conference

proceedings

# Third USENIX Workshop on Electronic Commerce Proceedings

*Boston, Massachusetts*
*August 31–September 3, 1998*

Sponsored by
**The USENIX Association**

USENIX®

The Advanced Computing
Systems Association

USENIX Third USENIX Workshop on Electronic Commerce *Boston, Massachusetts* *August/September 1998*

For additional copies of these proceedings contact:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710 USA
Phone: 510 528 8649
FAX: 510 548 5738
Email: office@usenix.org
WWW URL: http://www.usenix.org

The price is $20 for members and $26 for nonmembers.
Outside the U.S.A. and Canada, please add
$12 per copy for postage (via air printed matter).

**Past Electronic Commerce Proceedings**

| | | | |
|---|---|---|---|
| Second Electronic Commerce Workshop | November 1996 | Oakland, CA | $20/26 |
| First Electronic Commerce Workshop | July 1995 | New York, NY | $20/26 |

ISBN 1-880446-97-9

Printed in the United States of America on 50% recycled paper, 10-15% post consumer waste.

USENIX Association

# Proceedings of the

# 3rd USENIX Workshop on Electronic Commerce

August 31-September 3, 1998
Boston, Massachusetts

## Workshop Organizers

### Program Chair
Bennet Yee, *University of California, San Diego*

### Public Key Infrastructure Coordinator
Daniel Geer, *CertCo, LLC*

### Program Committee
Ross Anderson, *Cambridge University*
Marc Donner, *Morgan Stanley*
Niels Ferguson, *Digicash*
Mark Manasse, *Digital Equipment Corporation*
Clifford Neuman, *University of Southern California*
Avi Rubin, *AT&T Labs – Research*
Win Treese, *OpenMarket, Inc.*
Doug Tygar, *Carnegie Mellon University*
Hal Varian, *University of California, Berkeley*

### External Reviewer
*Terence Kelly, University of Michigan*

### The USENIX Association Staff

# Table of Contents

## 3rd USENIX Workshop on Electronic Commerce

## August 31-September 3, 1998
## Boston, Massachusetts

### Tuesday, September 1

**Advances in Payment Technology**
*Session Chair: Clifford Neuman, University of Southern California*

**Auction Markets**
*Session Chair: Hal Varian, University of California, Berkeley*

### Wednesday, September 2

**Secure Systems – What It Takes**
*Session Chair: Ross Anderson, Cambridge University*

### Deployable Internet/Web Services

*Session Chair: Avi Rubin, AT&T Labs – Research*

## Thursday, September 3

### Consumer Service

*Session Chair: Win Treese, OpenMarket Inc.*

### Short Talks/Works-in-Progress Reports

*Session Chair: Marc Donner, Morgan Stanley*

## Appendix

# Preface

Electronic commerce has grown tremendously since these workshops began in 1995. Recent U.S. government figures show that Internet transactions totalled $8 billion last year and project that by 2002 the total will likely surpass $300 billion. Industry sources estimate that business-to-business electronic commerce will total $16 billion this year, with over $8 billion of that in the form of business-to-business auctions.

We are living in interesting times — not only is the volume of electronic commerce undergoing tremendous growth, but new electronic commerce ideas and applications are also constantly being developed. Arising hand-in-hand with these are new issues, ranging from immediately practical ones, such as scalability and on-line privacy, to more abstract ones like efficient market design. The USENIX Electronic Commerce Workshop continues to be a scholarly, scientific forum where these ideas and issues are explored.

In this year's workshop, we have 16 refereed technical papers (out of 30 submissions). We hope to also prepare a digest of the meeting consisting of brief summaries of the talks, questions, and discussions to be made available later. Such a digest from the last Workshop is included as an appendix to this year's proceedings.

I sincerely thank the program committee for taking the time from their busy schedules to carefully review the papers, and the USENIX staff for their excellent work in putting this workshop together. Thanks also to Dan Geer for putting together the PKI track, and to Peter Honeyman for invaluable advice.

Bennet Yee<br>
Program Chair

# Electronic Commerce and the Street Performer Protocol

John Kelsey Bruce Schneier
{kelsey,schneier}@counterpane.com
Counterpane Systems, 101 East Minnehaha Parkway, Minneapolis, MN 55419

## Abstract

We present two software-network payment systems, designed so that every user is capable of both buying and selling. One system uses online clearing; the other uses offline clearing.

## 1 Introduction

Internet commerce requires new payment methods, and several have already been developed [OO90, OO92, FY92, Bra93a, Bra93b, MN93, CR93, Fer94, LMP94, Oka95, MN95, BGH+95, ST95, TMSW95]. This paper describes a specific kind of payment system in which each user can transfer money to each other user: a peer-to-peer payment system. This can be thought of as a credit card account or a checking account. At regular intervals, these electronic payments are translated into real-world payments. Although this step is important to the users of these protocols, it's not a cryptographic operation at all; just a matter of executing EFTs or writing and mailing out checks.

This paper is organized as follows. Section 2 is a discussion of the objectives and design principles for this kind of system. In Section 3 we describe two example systems, one using online clearing, the other, offline clearing with public key signatures. Neither of these is meant as a final system, but both serve as good outlines to build a working system around.

## 2 Objectives, Design Principles, and Options

The ultimate objective is to have a payment system as handy for doing internet trade as cash, checks, or credit cards are for doing day-to-day trade. Whatever this is, it must meet the following criteria:

1. *Secure.* The system must be secure in the sense that it must not be possible to convince someone they have received money when they have not. It must also not be possible to forge a payment from someone else, or to replay payments. Finally, it mustn't be possible to violate other design specifications of the system, for example by tracing other peoples' payments.

2. *Cheap.* The system shouldn't require the user to purchase expensive equipment (other than a PC). The operating costs of the system should be low enough that nobody has to pay a large fixed monthly fee. Indeed, a small fixed monthly fee may be all anyone pays, if operation is cheap enough. In a computational sense, nobody should have to do too many complicated computations to pay someone a dime.

3. *Widely Available.* For maximum availability, the client (peer) software should be partially or completely available in source-code-- participation in the payment system might be the paid for monthly, along with settling of bills as needed.

4. *Peer-to-Peer.* Each user should be able to either send or receive payments without complicated registration procedures.

There are several tradeoffs. For example, making a payment protocol computationally light usually means using few or no public key operations. However, this seems to require some kind of interaction with a central server, which drives up communications requirements and operating costs.

In this paper, we attempt to stay within the constraints of what is implementable in software on standard computers of today, using a good cryptographic function library. Individual users (peers) may have secrets, but these secrets have relevance only to those users. Only the bank is allowed to hold any global secrets.

### 2.1 Notation

Notation in the remainder of this document is as follows:

- $Enc_{K_A}(X)$ means that $X$ is encrypted under key $K_A$, using some symmetric encryption algorithm.
- $Auth_{K_A}(X)$ means that $X$ is authenticated under key $K_A$, using some symmetric message authentication algorithm.
- $EncAuth_{K_A}(X)$ means that $X$ is authenticated and encrypted, using symmetric algorithms, under keys derived from $K_A$.
- $PKEnc_{K_A}(X)$ means that $X$ is encrypted under public key $K_A$.
- $Sign_{K_A}(X)$ means that $X$ is digitally signed under private key $K_A$.
- $X, Y$ means that $X$ is concatenated with $Y$.

## 3 Online Clearing Systems

Online clearing systems are simpler than offline systems, at least in principle, since both users can authenticate themselves to the Bank's trusted server instead of to each other. There are no certificates to worry about, and the most important security operations take place on a secured machine. Also, this system obviates the need for computationally expensive public-key operations (and the associated licensing issues). The cost of this is that the Bank has to maintain a constant online presence, and can easily become the bottleneck of the system.

In this section, we briefly discuss one example of an online system. In this system, we require the person accepting the payment to have a reasonably accurate clock, and to keep some short-term memory about transfers that have been accepted recently, to prevent trivial replay attacks. When Alice wants to transfer money to Carol, she first gets an electronic note of approval from the Bank for this transfer, which includes an expiration time (after which Carol should not accept it). Then, she sends this note to Carol to convince her that the transfer has indeed taken place. This turns out to be very close to the Kerberos protocol [Sch96].

### 3.1 Variables

1. $S_A$ and $S_C$ are Alice and Carol's sequence numbers. These must never repeat or go backwards—instead, they increment one value at a time. Suspicious jumping around by the sequence number must always be noted by the bank, and should initiate corrective action of some kind.
2. $ID_A$ and $ID_C$ are Alice and Carol's unique 64-bit ID numbers.
3. $K_A$ and $K_C$ are the keys shared between Alice and the Bank, and Carol and the Bank, respectively. The keys are derived based on a master key held by the bank, $K_*$, according to the relation
$$K_i = E_{K_*}(ID_i)$$
4. The audit-log is the log of all previous payments, kept by Alice's software. By including this hash, Alice commits to the current entries in the log; if she alters these later, it will be detected by the Bank [SK96].
5. A Timestamp is a 32-bit representation of a given time and date.

### 3.2 Basic Protocol: Alice Transfers Money to Carol

1. Alice forms

   $U_0 =$ "Request for Transfer Authorization

   $S_A =$ Current sequence number for Alice–incremented immediately

   $V = hash$(Audit Log)

   $W =$ Amount of Transfer

   $X_0 = U_0, S_A, V, W, ID_C$

   $K_0 =$ a random message key

   and sends to the Bank

   $M_0 = ID_A, EncAuth_{K_A}(K_0), EncAuth_{K_0}(X_0)$

2. The Bank receives and decrypts this. If the message doesn't decrypt or authenticate properly, the Bank responds with a simple error message. If the sequence number or hash of the audit log is wrong, then it must begin corrective action. This will be discussed further below. If the amount of the transfer is more than the

amount in Alice's account, the ID for Carol is invalid, or anything else is wrong with the message, then a simple error message is sent to Alice. However, if nothing is wrong with $M_0$, then the Bank forms

$U_{1a} =$ "Transfer Authorization"

$T_E =$ Expiration time of authorization

$Y_1 = U_{1a}, hash(M_0) ID_A, ID_C, W, T_E$

$K_1 =$ a random message key

$Z_1 = EncAuth_{K_C}(K_1), EncAuth_{K_1}(Y_1)$

$U_{1b} =$ "Transfer Verification"

$S_C =$ Carol's current sequence number."

$ID_A, ID_C, W$ from above step.

$X_1 = U_{1b}$, $hash(M_0)$, $ID_A$, $ID_C$, $W$, $S_C$, $T_E$

$K_2 =$ a random message key

and sends back to Alice

$M_1 = EncAuth_{K_A}(K_2), EncAuth_{K_2}(X_1, Z_1)$

3. Alice receives this, and verifies everything she can verify. If there are no problems, she forms

   $U_2 =$ "Payment"

   $Z_1$ from above.

   and sends to Carol

   $M_2 = U_2, Z_1$

   At this point, Alice updates her audit log to include this transaction. This is done by appending $ID_C, W$ to the previous contents of the audit log. In the event of any trouble verifying that this payment arrived, Alice and/or Carol contact the Bank.

4. Carol appends $ID_A, W$ to her audit log, and adjusts her internally-carried balance by the amount of the transfer.

### 3.2.1 Security of the Protocol

1. *Replays*

   (a) The Bank would recognize any replay attempts immediately because of Alice's sequence number. Only an attacker that could alter or forge messages from Alice could replay a message with a new sequence number, without being caught at it.

   (b) Alice would recognize any replay attempts because the hash of her previous message is embedded inside the second protocol message.

   (c) Carol would recognize replay attempts directed at her by the replayed timestamp and incorrect sequence number. Note that there needs to be some room for error in both these values for Carol, because messages may not always arrive in the order they were sent, and because most computer clocks aren't highly accurate.

   (d) It's not possible to alter individual parts of transmitted messages because of the authentication being used. It's not possible to alter individual messages in a protocol because they typically contain the hash of the prior message.

2. *Forgeries* Forgeries should be very difficult to carry out, if the symmetric authentication scheme is acceptably strong.

3. *Information Leaks* Because of the use of a new message key for each message, it should be very hard for any information to leak. Within the message formats, only the ID of the person transferring money to someone else is leaked in any message.

4. *Other Attacks* There is a trivial attack in which Carol simply discards a payment to her by Alice, which means that Alice and the Bank will eventually need to resolve this. This is fundamentally a nuisance attack.

## 3.3 Secondary Protocol: Alice Reconciles

Occasionally, communications failures, implementation errors, or fraudulent transfers may leave Alice and the Bank unsynchronized. The solution is to go through a more complicated protocol to verify that all is well. This is also done from time to time to verify that Alice has received all the deposits that she should have received by now. During the ordinary transfer protocol, the Bank can require a reconciliation protocol before it will allow Alice to carry out any transfers, or Carol to receive any. This protocol is always required before depositing or withdrawing any money from the bank.

1. Alice forms

$U_0$ = "Request for Reconciliation"

$R_0$ = a random challenge

$S_A$ = Alice's current internal sequence number

$X_0 = U_0, S_A$

$K_0$ = A random message key

she then sends to the Bank

$M_0 = ID_A, EncAuth_{K_A}(K_0), EncAuth_{K_0}(X_0)$.

2. The Bank verifies the authentication, and then forms

   $U_1$ = "Reconciliation Challenge"

   $S^*_A$ = Bank's current idea about what $S_A$ should be.

   $K_1$ = A random message key

   $X_1 = U_1, hash(M_0), S^*_A$

   it then sends to Alice

   $M_1 = EncAuth_{K_A}(K_1), EncAuth_{K_1}(X_1)$.

3. Alice verifies the authentication, and that the hash of the previous message is included. She then forms

   $U_2$ = "Reconciliation Response"

   $X_2 = U_2, hash(M_1)$,Full Authentication Log

   $K_2$ = a random message key

   $S_A = max(S_A, S^*_A) + 1$

   she then sends to the Bank

   $M_2 = EncAuth_{K_A}(K_2), EncAuth_{K_2}(X_2)$.

4. The Bank, after verifying all this information, either is or is not willing to let Alice start transferring money again. Thus, it forms

   $U_{3a}$ = "Reconciliation Success Message"

   $U_{3b}$ = "Reconcilliation Failure Message"

   $K_3$ = a random message key

   $X_{3a} = U_{3a}$, $hash(M_2)$, $S_A$, Account Balance

   $X_{3b} = U{+}3b$, $hash(M_2)$, Problem Description

   and sends to Alice, depending on the circumstances, one of the following:

   $M_{3a} = EncAuth_{K_A}(K_3), EncAuth_{K_3}(X_{3a})$.

   $M_{3b} = EncAuth_{K_A}(K_3), EncAuth_{K_3}(X_{3b})$.

At the end of this protocol, Alice and the Bank should have some common new sequence number, and either they should agree on what has happened, or there will be some kind of investigation going on, and Alice should know the basic kind of problem and what to do next.

### 3.4 What Can Go Wrong?

The most dire failure would involve a security breach at the Bank. Maintaining an authenticated, physically secure and separate log would be one way to ensure that a bank could recover from this kind of failure. Note that each payment from a user contains a current hash of her log.

The user's PC can lose security, in almost the same sense that someone can lose a credit card or checkbook. This should be rare, but it will generally have to be dealt with by humans. The software maintains a log, authenticated by chained hashes. Each payment commits to the current value of the log; that is, the hash of the log up until now.

The authentication and/or encryption functions can be broken. If this happens, the system must be recalled and fixed, and lots of money will be lost. This gives us a lot of incentive to choose our authentication and encryption functions well. For example, Triple-DES [NBS77] or Blowfish [Sch94] in CBC-mode might be used to encrypt, and a keyed hash might be made from SHA1 [NIST93], using a well-thought-out construction such as [PvO95].

## 4 An Off-line Payment System

The offline system can be imagined as a checking account: users are allowed to write checks for whatever's in their accounts. The software will not allow them to overdraw, but it is assumed that fraudulent users can change their software to allow this. We assume that we know how to deal with people who don't pay their bills: they can overdraw for a limited span of time, and the bank's collections department will wind up trying to get the money back from them. The offline system uses some public key operations instead of interactions with the

Bank. Although certificates are used, there is no CRL. Instead, certificates have a very short lifetime, perhaps a week. After that time, the user must connect to the bank to get a new certificate.

There are two routine operations: Payment and Reconciliation. In payment, one user (Alice) will pay $\$X$ to another user (Carol). In reconciliation, a user (Alice) will interact with the Bank to allow her to continue using the system. The system assumes that both Alice and Carol have reasonably accurate clocks.

## 4.1 Payment

In this protocol, Alice makes a payment to Carol by sending Carol a "draft." That is, a timestamped, digitally signed authorization for the bank to move $\$X$ from Alice's account into Carol's. Each draft has a unique identifying number, which prevents the bank from ever accepting replays, and gives Alice a way to refer to disputed payments.

Variables:

1. Timestamp is the current timestamp, in some standard format. The timestamp must never repeat, so it should include everything from centuries to seconds. An example might be "19951225010000".
2. The audit-log is the log of all previous payments. By including this, Alice commits to the current contents of her log each time she makes a payment. Later changes are detected by the Bank during reconciliation.
3. The Draft is the order specifying a draft sequence number, the account number of the payor, and the account number of the payee.
4. $Cert_A$ is Alice's certificate, attesting to the current valid linkage between her public signing key and her ability to write drafts on a given account id.
5. $PK_A$ is Alice's public signing key.

This is the protocol by which Alice pays Carol:

1. Alice forms

   $U_0 =$ "Check Transmission"

   $T_0 =$ timestamp

   $V_0 = hash(audit - log)$

   $X_0 = U_0, V_0, T_0,$ Draft

   $S_0 = Sign_{SK_A}(X_0)$

   $K_0 =$ a random message key

   and sends to Carol

   $M_1 = Cert_A, PKE_{PK_C}(K_0), Enc_{K_0}(X_0, S_0)$

2. Carol verifies Alice's certificate and timestamp. She then verifies the signature on Alice's draft. If all is well, she knows that the payment has occurred.

This protocol is somewhat computationally expensive. Alice must perform a signature, and Carol must perform two verifications (though she may want to maintain lists of valid certificates, so she doesn't perform the certificate verification more often than necessary). Note that all payments from Alice to Carol appear in the clear; if a secure connection has already been made, then this will work well. If not, it may be necessary in some cases to establish a secure connection; this implies more public key operations for a strong key exchange. Also note that nothing keeps Carol from refusing to acknowledge Alice's payment; Alice can contact the bank and send a signed request to have a stop-payment put on that draft.

## 4.2 Deposit

Deposit simply consists of Carol sending to the Bank (perhaps by e-mail) the digitally signed draft from Alice. Since the draft names Carol, there is no risk of redirection. An active attack can prevent the bank from receiving the deposit. The Bank returns a receipt, also digitally signed. Because each draft must be different, there is no risk of replay attack, if things are done properly.

This is the protocol by which Carol deposits a draft:

1. Carol forms

   $U_1 =$ "Deposit Request"

   $T = Timestamp$

   $V_1 = X_0, S_0$ from the previous protocol

   $X_1 = U_1, T, V_1.$

   $S_1 = Sign_{SK_C}(X_1)$

   $K_1 =$ a random message key

and sends to the Bank

$$M_1 = PKE_{PK_C}(K_1), Enc_{K_1}(X_1, S_1)$$

2. The Bank verifies that all is well. Tf so, it forms

   $U_2 =$ "Deposit Receipt"

   $R = Receipt$

   $X_2 = U_2, R, hash(M_1)$

   $S_2 = Sign_{SK_B}(X_2)$

   $K_2 =$ a random message key

   and sends to Carol

   $$M_2 = PKE_{PK_C}(K_2), Enc_{K_2}(X_2, S_2)$$

If Carol never receives $M_1$, she restarts the protocol later, or eventually notes it when she reconciles.

## 4.3 Reconciliation

Reconciliation occurs when Alice connects with the Bank to send in her logs, including copies of all drafts she has written, and to receive a new certificate. User certificates expire often.

1. The Bank forms

   $U_3 =$ "Reconcilliation Request"

   $R_3 =$ a random 64-bit value

   $X_3 = U_3, R_3$

   $S_3 = Sign_{SK_B}(X_3)$

   $K_3 =$ a random message key

   and sends to Alice

   $$M_3 = PKE_{PK_A}(K_3), Enc_{K_3}(X_3, S_3)$$

2. Alice forms

   $R_4 =$ a random 64-bit value

   $U_4 =$ "Reconcilliation Response"

   $L =$ the log of payments and deposits since last reconciliation, noting any for which she received no receipt, and any for which she wishes to dispute payment.

   $X_4 = U_4, hash(M_3), R_4, L$

   $S_4 = Sign_{SK_A}(X_4)$

   $K_4 =$ a random message key

   and sends to the Bank

   $$M_4 = PKE_{PK_B}(K_4), Enc_{K_4}(X_4, S_4).$$

3. The Bank verifies that all is well. If so, it responds by forming

   $U_5 =$ "Reconciliation Receipt"

   $C_A =$ New Certificate for Alice with a new timestamp

   $X_5 = U_5, hash(M_4), C_A,$ Ending Balance, Timestamp

   $S_5 = Sign_{SK_B}(X_5)$

   $K_5 =$ a random message key

   and sends to Alice

   $$M_5 = PKE_{PK_A}(K_5), E_{K_5}(X_5, S_5)$$

If there are disputed payments or other problems, there will be a request for more information sent. This should lead to an online session, telephone call, or face-to-face discussion of the user's complaint. Dispute resolution falls outside of the scope of the payment system.

## 4.4 Error Recovery, Fraud Detection, and Auditing

A continual audit trail is kept in the user's PC, and its current value is committed to each time a payment is made. Forged transactions will be caught during reconciliation, as will almost all other problems. People who overdraw their accounts, exceed their credit lines, or fail to pay their bills have either had a software flaw, or hacked their software to do these things. They will not be issued a new certificate until they resolve the problem. This limits damage from individual security failures to a small period of time, perhaps no more than a week.

## 4.5 What Can Go Wrong?

Compromise of the private signing key used to create certificates is probably the most catastrophic thing that could happen. This must be guarded against strongly, perhaps maintaining the key in a tamper-resistant token under good physical security. Compromise of this key would allow an attacker to write an endless stream of bad checks on anyone's account.

The user's PC can lose security, in almost the same sense that someone can lose a credit card or checkbook. This should be rare, but it will generally have to be dealt with by humans. The software maintains a log, authenticated by chained hashes. Each payment commits to the current value of the log: i.e., the hash of the log up until now. Since certificates expire often, this will be dealt with soon after it's discovered by freezing the account that has been compromised until the problem can be dealt with. In this system, another thing that can happen is that a user's clock can be fouled up, to trick him into accepting a stale certificate. Such attacks must be guarded against.

The authentication, signing, and/or encryption functions can be broken. If this happens, the system must be recalled and fixed, and lots of money will be lost. This gives us a lot of incentive to choose our functions well. For example, we might use 1280-bit RSA [RSA78] SHA1 [NIST93] for our signatures.

### 4.6 Additional Comments and Extensions

Note that it is possible to give the payor anonymity from the payee. This is done by giving each user a large number of IDs, each with a different public signing key and certificate. This gives no anonymity from the bank.

## 5 Retrofitting Existing Systems

There are several reasonably good methods of paying for things available now. Small variations in these could be adapted to this kind of an applications. The advantages of using an already fielded system are that there is no need to build the system's infrastructure, and that a fielded system may already have had many of its flaws worked out. Among the possible disadvantages are that many currently-fielded payments systems don't meet the specific needs of this application, and that there are often hardware requirements and licensing fees.

- Digicash is payer-anonymous electronic cash. It is currently available. The cost per transaction appears to be too high, and there are some practical security problems with any payer-anonymous system that need to be worked out, but this kind of system might be worth using for this application at some point.
- Some schemes simply transfer users' credit card numbers around. These have obvious security problems involving replay and forgery attacks. However, one way to implement any of the systems discussed above would be for the bank to bill users' credit card for any money owed. To avoid high transaction costs, the bank would bundle many small transactions per month before billing them—perhaps only in $50 increments—from the user's card. The advantage of this would be in low start-up costs. Ideally, the system would also be able to credit money to their credit cards, or transfer it into their bank accounts. This would simplify the mechanics of running this kind of system.
- There are micropayment schemes, such as Millicent, Payword, and Micromint, that allow for small monetary values to be transferred among parties. However, micropayment schemes generally have to solve somewhat different problems than a peer-to-peer metering system, so they make somewhat different tradeoffs.

## 6 Conclusion

The future of the Internet is one where many people are both producers and consumers of information. Any payment system needs to reflect this. This work, while no means complete, shows the kind of directions necessary for peer-to-peer payment systems.

## References

[BGH+95] M. Bellare, J.A. Garay, R. Hauser, A. Herzberg, H, Krawczyk, M. Steiner, G. Tsudik, and M. Waidner, "iKP - A Family of Secure Electronic Payment Protocols", *The First USENIX Workshop on Electronic Commerce,* USENIX Association, 1995, pp. 89–106.

[Bra93a] S. Brands, "Untraceable Off-line Cash in Wallets with Observers," *Advances in Cryptology—CRYPTO '93 Proceedings,* Springer-Verlag, 1994 pp. 302–318.

[Bra93b] S. Brands, "An Efficient Off-line Electronic Cash Systems Based on the Representation Problem," C.W.I. Technical Report CS-T9323, 1993.

[CR93] CitiBank and S. S. Rosen, “Electronic-Monetary System,” International Publication Number WO 93/10503; May 27 1993.

[Fer94] N. Ferguson, “Extensions of Single-Term Coins,” *Advances in Cryptology—CRYPTO ’93 Proceedings*, Springer-Verlag, 1994, pp. 292–301.

[FY92] M. Franklin and M. Yung, “Towards Provably Secure Efficient Electronic Cash,” Columbia Univ. Dept of C.S. TR CUCS-018-92, April 24, 1992. (Also in Icalp-93, July 93, Lund Sweden, LNCS Springer-Verlag).

[LMP94] S. H. Low, N. F. Maxemchuk and S. Paul, “Anonymous Credit Cards,” *The Second ACM Conference on Computer and Communications Security,* ACM Press, 1994, pp. 108–117.

[MN93] G. Medvinsky and B. C. Neuman, “Netcash: A Design for Practical Electronic Currency on the Internet,” *The First ACM Conference on Computer and Communications Security,* ACM Press, 1993, pp. 102–106.

[NBS77] National Bureau of Standards, NBS FIPS PUB 46, “Data Encryption Standard,” National Bureau of Standards, U.S. Department of Commerce, Jan 1977.

[MN95] B. C. Neuman and G. Medvinsky, “Requirements for Network Payment: The NetCheque$^{TM}$ Perspective,” Compcon ’95, pp. 32–36

[NIST93] National Institute of Standards and Technology, NIST FIPS PUB 180, “Secure Hash Standard,” U.S. Department of Commerce, May 93.

[Oka95] T Okamoto, “An Efficient Divisible Electronic Cash Scheme,” *Advances in Cryptology—CRYPTO ’95 Proceedings*, Springer-Verlag, 1995, pp. 438–451.

[OO90] T. Okamoto and K. Ohta, “Disposable Zero-Knowledge Authentication and Their Applications to Untraceable Electronic Cash,” *Advances in Cryptology—CRYPTO ’89 Proceedings*, Springer-Verlag, 1990, pp. 481–496

[OO92] T. Okamoto and K. Ohta, “Universal Electronic Cash,” *Advances in Cryptology—CRYPTO ’91 Proceedings*, Springer-Verlag, 1992, pp. 324–337.

[PvO95] B. Preneel and P.C. van Oorschot, “MD$X$-MAC and Building Fast MACs from Hash Functions,” *Advances in Cryptology—CRYPTO ’95 Proceedings*, Springer-Verlag, 1995, pp. 1–14.

[RSA78] R. Rivest, A. Shamir, and L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” *Communications of the ACM*, v. 21, n. 2, Feb 1978, pp. 120-126.

[Sch94] B. Schneier, “Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish),” *Fast Software Encryption, Cambridge Security Workshop Proceedings*, Springer-Verlag, 1994, pp. 191–204.

[Sch96] B. Schneier, *Applied Cryptography, Second Edition*, John Wiley & Sons, 1996.

[SK96] B. Schneier, J. Kelsey, “Automatic Event-Stream Notarization Using Digital Signatures,” in *Advances in Cryptology, Proceedings of the Cambridge Protocols Workshop 96*, Springer-Verlag, 1996, pp. in preparation.

[ST95] M. Sirbu and J. D. Tygar, “NetBill: An Internet Commerce System Optimized for Network Delivered Services,” Compcon ’95, pp. 20–25.

[TMSW95] J. M. Tenenbaum, C. Medich, A. M. Schiffman, and W. T. Wong, “CommerceNet: Spontaneous Electronic Commerce on the Internet,” Compcon ’95, pp. 38–43

# *VarietyCash*: a Multi-purpose Electronic Payment System

(Extended Abstract)

M. Bellare[*] J. Garay[†] C. Jutla[‡] M. Yung[§]

## Abstract

*VarietyCash* is a new electronic money system which strikes a balance between functionality, security, and user privacy. The system can encompass both network and stored-value card based payment mechanisms, with transferability between them, hence the name.

## 1 Introduction

Electronic payment systems are well-understood to be an essential step on the road to electronic commerce, and are thus in high demand. These systems need to strike a good balance between a number of different issues. Amongst these are:

- *Anonymity:* The extent to which a third party (bank or other payment service provider) or the merchant has information about the identity of the buyer.
- *Account-based or account-less:* Account based systems are higher cost from the point of view of the service provider. In the US, at least, so-called Regulation E obligates the service provider of account-based systems to provide customers with a level of account maintenance (e.g., periodic statements) which increases the cost of such systems.
- *Network versus card-based:* Some systems are software-based for transactions across the network; in others, like Mondex [14], stored-value cards are used.
- *Atomicity:* The system should be fair and robust in the sense that network failures, for example, do not result in incomplete transactions [22].

We propose a simple, versatile system that is *account-less*, provides some degree of anonymity in the form of *anonymity by trust*, is versatile enough to allow both network-based and card-based payment to inter-operate in the system, and provides atomicity. The system is the result of a thorough system's analysis, design, and proof of concept carried out by the authors for a large financial institution.

### 1.1 Background

There are many good reasons to have some degree of anonymity: people want to avoid being added to mailing lists, or simply keep secret the kinds of goods they buy; businesses may not want competitors to know what kinds of information they are buying. The strongest form of anonymity (unconditional) is provided by systems based on so-called blind signatures [10]. (These systems were first proposed by Chaum [8]. At this point, there are many of them published.) However, these systems have many well-known drawbacks. Tygar and Yee (as reported in [22]) point out that the protocols lack atomicity: network failures during a transaction will result in loss of the electronic money involved. That double spending is only caught after the fact is considered too high risk by banks. Another problem (as pointed

[*] Department of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA. E-mail: `mihir@cs.ucsd.edu`. URL: `//www-cse.ucsd.edu/users/mihir`. Supported in part by a 1996 Packard Foundation Fellowship in Science and Engineering, and by NSF CAREER Award CCR-9624439.

[†] Information Sciences Research Center, Bell Laboratories, 600 Mountain Ave, Murray Hill, NJ 07974. E-mail: `garay@research.bell-labs.com`. URL: `www.bell-labs.com/user/garay/`.

[‡] IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, New York 10598. E-mail: `csjutla@watson.ibm.com`.

[§] CertCo, 280 Park Avenue, New York, NY 10017, USA. E-mail: `moti@certco.com`.

out in [16]) is scalability: as all coins that have been spent have to be recorded, the coin data base will grow over time, increasing the cost to detect double spending. Attempts to address some of these issues have been made (e.g. via escrow [13]) and stored-value card based settings mitigate others [7], but drawbacks remain.

Mondex [14], on the other side of the spectrum, provides no anonymity. It seems to be a shared key based card design. There are well-known scalability and security problems with such designs. For example the security is totally dependent on the security of a master key, and this master key must be distributed to many users and points of sale across the system. If a single module is penetrated, not only is significant retailer fraud facilitated, but the entire card base may be compromised.

NetBill [19] has a large spectrum of desirable properties, including atomicity. It might be able to provide some limited anonymity via trust in the NetBill server and the use of pseudonyms, but linkage is still possible. Also the NetBill server maintains accounts for buyers and merchants, which is costly. Also, NetBill's transaction protocols include functions that are beyond payment, such as price *negotiation.* The trend nowadays is that these functions should be separated, and standardized; e.g., JEPI [12], SEMPER [20].

Our design (its network version) shares many features with NetCash [16, 15] (NetCash has no "off-line" operation, meaning the ability to also work with stored-value cards). The NetCash protocols also combine symmetric-key cryptography for performance with public-key mechanisms. A consequence of shared-key operations is no non–repudiability in some key functions (e.g., in exchanges with currency server). Anonymity in NetCash is also trust-based, with multiple currency servers which the client selects; this in turn implies an accounting infrastructure in order to maintain consistency accross servers, etc. [17]. Additionally, NetCash is cast as a framework that can accommodate various electronic currency mechanisms,

*i*KP [3], SET [21], etc., are credit card-based payment systems. We are looking for some form of cash.

## 1.2 Variety Cash

In *VarietyCash*, the issuer functions as a mint. Coins are tokens authenticated under an issuer master key. The system is on-line, meaning the issuer maintains a coin database, and the merchant checks on-line with the issuer that a coin has not been previously spent. Since the system is on-line, the master key is present only at the issuer, so, unlike in Mondex, can be well protected.

Spent coins can be erased from the database, unlike some systems; if this were not possible scalability would be adversely impacted.

*VarietyCash* provides "trust-based anonymity." At withdrawal time the issuer does note the association of user ID to coin serial numbers, but this user database is separate from the database recording the spent or unspent status of a coin which is looked up when a merchant wants to confirm that no double spending has occurred. The information can be co-related as necessary in case of law enforcement needs, but the issuer is expected to maintain user privacy except in such cases. While not anonymity at the level of digicash, this relies on no more trust than we put in our financial service providers today, and with a well-known, reputable service provider, is likely to be deemed adequate by most people for most things. Furthermore, users can use pseudonyms.

The system is robust, functional and flexible. The protocols for withdrawal and spending will provide atomicity. Coins can be purchased in any number or denomination, and paid for in a variety of ways. The system is account-less, so Regulation E is avoided, and the payment service provider does not have to issue statements to customers or incur other such overhead. The same coins can be used for network based payment transactions or put on stored-value cards in the Mondex style, and the coins are transferable between the two. Novel features include transferability of coins without any specific payment transaction, ie. "making change".

The main concern is cost arising from it being an on-line system. The issuer will have to invest some resources to be able to handle lots of transactions quickly. This, however, appears feasible for reasonable load. Our protocols minimize the overhead. Batching and aggregation can be used to reduce costs (e.g., [4]).

This design originated in response to a request of a certain large financial service corporation to seek a practical, viable e-money system. In discussions with them we found they were more ready to take on the cost of on-line verification than to incur the risks arising from anonymous cash, or to fall under Regulation E. From this it appears that there is a market for a system like *VarietyCash.*

## 1.3 Implementation

Our design has been implemented in C++. The core of the system can be viewed as a distributed cryptographic application framework. The issuer and merchant sides have been implemented to run on AIX, whereas the client (buyer) runs on Windows 95. A buyer can access web-sites of merchants through a browser. If the web-site is e-money-enabled, on a purchase it returns a document with a special mime-type, which the browser has been configured to recognize. The browser then invokes an external e-money daemon. The daemon launches (if not already running) a graphical user interface for the client, and rest of the communication between the three parties takes place via the daemon. The transaction databases have been implemented persistently, so that malfunctions such as network breakdowns do not hamper the protocol.

### 1.4 Organization of the paper

We start in Section 2 by describing the model and system architecture of the main component of *VarietyCash*, namely, a network-based, on-line payment system. Some of the design considerations and requirements for the processes we present may be of wider applicability. In Section 3 we concentrate on two processes (and corresponding protocols) in detail: withdrawal of e-money and payment. Finally, in Section 4 we show how to integrate to the network component a typical card-based payment system.

## 2 Model and System Architecture

### 2.1 Security design goals

We start by identifying the basic aspects that compose the security of an e-money system.

PROTOCOL SECURITY. By this we mean liveness and safety guarantees, namely, that the protocols achieve their goals and that every participant gets its information, and is secure in the sense that the other parties which are considered adversaries do not compromise or spoil the system. This aspect is the main focus of this paper.

INTERNAL SECURITY. The security of the internal operation system of the issuer of electronic currency, its capability to withstand insider attacks and abuses. The internal network architecture, operation policies, employment of tamper-proof hardware as well as dual control measures and access-control and physical access limitations should be reviewed. The internal security architecture has to be combined with issues such as availability, reliability, load balancing and back-up requirements.

NETWORK SECURITY. The security of the network (e.g., Internet) of users and the issuer, to prevent attacks not via the protocol but rather through "break-ins;" these attacks exploit the lack of proper protection into the system and software holes. Careful design of the interface to the external network (firewall protection) is required. Both the internal and the network systems have to be evaluated under "Global Security Testing," which includes penetration attempts and security assessment of design and implementation.

USER SECURITY. Security of the user's assets. The user must obviously protect his electronic currency, and the software and procedures supplied to the user have to provide for protection at a proper level (e.g., beyond password-only protection), but at the same time be user-friendly.

In this paper, we deal specifically with the protocol aspects and their security. In this presentation we do not cover all the protocols, but what we cover seems to capture the basic needs of the system. For simplicity, nor do we deal with the temporal aspects of maintaining the system, such as long-term key management and cryptographic policies.

### 2.2 Parties and roles

We will use the following terminology for the *parties* involved in *VarietyCash*. In a typical transaction there would be three parties involved: the payer, the payee, and the bank or e-money server. However, we would like to be more specific than that:

- Participant – A generic name for a party involved in the system.
- Issuer ($I$) – Party who issues coins.
- Coin-holder ($CH$) – A party who holds coins or has the potential to do so.
- Payee ($PY$) – A party who is willing to accept coins as payment (e.g., a merchant).
- Bank ($B$) – A number of banks will be involved in moving funds due to conversions between electronic and real money.
- Certification Authority ($CA$) – The party who can "certify" the public keys of the participants. (In some scenarios, this role can be played by the Issuer itself.)

In turn, a coin-holder may play the following **roles**:

- Coin Purchaser ($CP$) – purchases coins from issuer
- Redeemer ($RD$) – turns coins into real money

- Payer ($PA$)– customer who pays for goods/services with coins
- Refresher ($Rf$) – gets new coins for old
- Changer ($Ch$) – makes change

Further, we make the following distinction amongst participants:

- Registerer ($RG$) – Register a public key at the issuer
- Enroller ($EN$) – Enroll for a particular role such as coin purchaser or merchant

The idea here is that a participant may wish to be able to "play the game," i.e., accept and use coins as a form of payment, without going through the more elaborate process that enables the purchase or redemption of e-money. Note that parties and roles are not mutually exclusive.

In *VarietyCash* all the participants have distinct identities, as well as public keys. The issuer has a database listing the identities of all participants, and, for each one, its public key. This information is entered at the time of registration. In all transactions directly involving the issuer, there is thus no need for an external certification authority; however, the design leaves an option for such a case.

We assume in this extended abstract a unique issuer, thus dispensing with the presentation of a more elaborate clearing system.

## 2.3 Adversaries and attacks

We distinguish between two major kinds of attackers: the *abusers* and the *spoilers*. Abusers try to get some specific advantage, such as get coins without properly paying for them; forge or steal coins; etc. Spoilers do not seek any advantage *per se*; they just try to disrupt the system. An example of a spoiler attack is to replay a coin purchase request in an attempt to make an unnecessary transfer of funds (from the withdrawer's bank to the issuer). Another example is the *denial of service* attack in which the attacker tries to tie up issuer resources.

Attackers may be *external* (e.g., on the Internet lines), or they may be parties themselves (for example, a malicious payee or withdrawer trying to get some money for free), or they may be *insiders* (such as an employee at the issuer).

Active attackers are the main problem. Pure eavesdroppers only try to learn information, such as the nature of a transaction. Our protocols will be designed to resist active attacks.

We do not discuss error handling or denial of service attacks. An adversary can always interrupt a flow and thus disrupt a protocol. It is assumed that standard re-transmission and time-out procedures underly the transmission of protocol flows and address these attacks as well as possible.

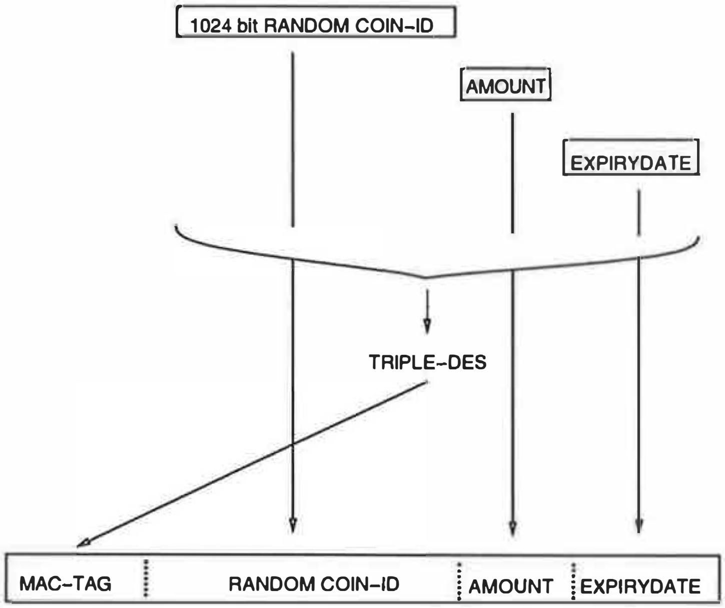
## 2.4 Coins and cryptographic terminology

An *e-coin* (*coin* for short) is an object consisting of a unique identifier (serial number, counter) called the *coin ID*, an indication of value (denomination), an expiry date, and an authenticating cryptographic tag. This cryptographic tag can be implemented in different ways. In our design we choose to use a MAC, or message authentication code, on the rest of the information, computed using a symmetric key which is kept in secret by the issuer. Thus, the issuer can compute and verify the tag, but no other party can compute a valid tag. Because it is secret-key based, the tag cannot even be checked by anyone except the issuer, i.e., this is not a digital signature. This is done for efficiency reasons—symmetric key based cryptography is hundreds of times more efficient than public-key operations. A schematic representation of a coin is shown in Figure 1.

The cost of successful tag forgery can be enormous, since if an adversary could forge tags, he or she could manufacture counterfeit coins at will. In order to minimize the possibility of successful tag forgery, we suggest the following:

- First, the tag should be computed in protected, tamper-proof hardware. This minimizes the risk of loss of the secret key.
- Second, the tag-computing algorithm should be "strong." One can of course use MACs based on existing primitives such as DES. This may be acceptable, but we suggest that a Triple-DES based MAC be used. A good choice would be a Wegman-Carter [23] type MAC, meaning that one applies an XOR-universal hash function to the data and then XORs the result with the value of a Triple-DES based pseudorandom function applied to a counter. The size of each tag should be (at least) 128 bits, so the total tag length is 28 bytes. Outside the cryptographic module the tag is encrypted inside the issuer database, and it goes encrypted over the network. It is only available in plain form to the receiver of the coin.

Additional protection is provided by the fact that the coin database itself is protected. So that if an adversary manages to forge the correct tag for a coin which has not yet been issued, it will not help, because the issuer will fail to validate the coin. Thus,



TRIPLE-DES MAC-TAG above is computed using a secret key K1 of the mint.

Figure 1: *Structure of* an un-*encrypted coin.*

the adversary must be able to successfully forge tags of issued, un-spent coins to achieve a meaningful gain.

A coin can have various *states.* For example, spent or not; anonymous or not; split, and if so how; etc. These are marked in the database.

The issuer will expect from a coin purchaser requesting coins, or a change maker wanting change, a specification of exactly what kinds of coins are being requested. The specification takes the form of a list of denominations, and for each denomination, the number of coins of that denomination that is desired. The simplest thing is to just ask for one coin of a certain denomination, for example a single coin of $0.80. But one could ask for, say, $(2, \$2.50), (1, \$1.25), (3, \$2)$, meaning I want 2 coins of value $2.50 each, one coin of value $1.25, and 3 coins of value $2. The *total value* of the list is the total dollar value, $12.25 in the example just given. The choice of specification is decided by a combination of the user's needs and the software (*purse*).

The cryptographic primitives used in the protocols of Section 3 are summarized in Figure 2. All the parties have public keys. The issuer has a cache of identities and their corresponding public keys, so that the certification authority is not needed in transactions with the issuer. (But it may be needed for other transactions.)

The encryption function $\mathbf{E}^*_X$ must provide, besides secrecy, some form of "message integrity." Decryption of a ciphertext results either in a plaintext message, or in a flag indicating non-validity. Formally, the property we require of the encryption scheme is *plaintext awareness* [5, 2]. Roughly speaking, this means that correct decryption convinces the decryptor that the transmitter "knows" the plaintext that was encrypted. This is the strongest known type of security, and in particular it is shown in [2] that it implies non-malleability (it is not possible to modify a ciphertext in such a way that the resulting plaintext is meaningfully related to the original one, as formalized in [11]) and security against chosen-ciphertext attacks. A simple, efficient scheme to achieve such encryption using RSA is OAEP [5]. A Diffie-Hellman based solution can be found in [1]. (Note that the RSA PKCS #1 encryption standard can be broken under chosen ciphertext attacks [6], and is thus not suitable for our purposes.)

However we stress that plaintext-aware encryption does not provide authentication in the manner of a signature, i.e., it does not provide non-repudiation. But it prevents an adversary from tampering with a

- **Keys:**

| | |
|---|---|
| $\mathrm{PK}_X, \mathrm{SK}_X$ | Public and secret key of Party $X$ |
| $\mathrm{CERT}_X$ | Public key certificate of Party $X$, issued by $CA$. We assume it includes $X, \mathrm{PK}_X$ and $CA$'s signature on $\mathrm{PK}_X$. |

- **Cryptographic primitives:**

| | |
|---|---|
| $\mathcal{H}(\cdot)$ | A strong collision-resistant one-way hash function. Think of $\mathcal{H}(\cdot)$ as returning "random" values. |
| $\mathbf{E}^*_X$ | Plaintext-aware public key encryption using $\mathrm{PK}_X$ |
| $\mathbf{S}_X(\cdot)$ | Digital signature with respect to $\mathrm{SK}_X$. Note the signature of message $M$ does NOT include $M$. We assume the signature function hashes the message before signing. |
| $\mathbf{e}_K$ | Symmetric key based encryption algorithm, taking key $K$ and a plaintext, and producing the ciphertext |
| $\mathbf{mac}_K$ | Symmetric key based signature, or message authentication code (MAC), taking key $K$ and a plaintext, and returning a short tag. |

Figure 2: *Keys and cryptographic primitives used in protocols*

ciphertext.

Also note that the encryption function is *randomized:* $\mathbf{E}^*$, invoked upon message $m$ will use, to compute its output, some randomizer, so that each encryption is different from previous ones.

Finally, the notation $\oplus$ denotes bitwise XOR.

## 2.5 Databases, modules, components

We briefly mention the components that interact with the processes we describe. Briefly, a participant has a purse, which has a database of coins. The issuer has a coin database and a participant database. These indicate the status of a coin, including the withdrawer ID if anonymity was not requested. A schematic view of the issuer's coin database is shown in Figure 3.

The issuer has a secure cryptographic module for coin generation. The coin-generation key is hardware-protected inside this module.

## 2.6 General requirements and principles

A global requirement is the *conservation of cash.* This means that the total e-money in the system is equal to the total amount of real money that the issuer's logs show is in e-money.

A general principle is that any coin representation is seen by at most one participant other than the issuer. Thus, after an issuer issues a coin, the withdrawer is the only party who "sees" this coin. When the withdrawer makes a payment with it, the payee doesn't see the coin; it is in a digitally sealed envelope which goes straight to the issuer for validation. This implies "on-line" payments where the issuer is involved in every such transaction. In addition, the internal representation of the coin does not enable insiders with access to its database to use it—a coin is checked by a tamper-proof hardware for its validity.

Coins are treated as bearer instruments, like real currency. The user has the money if s/he has a (valid) coin; no questions asked.

The security requirements that we pursue are those of "strong cryptography" for the protection of financial transactions; the use of "weak cryptography" only (e.g., 40 bit-long keys and passwords) is insufficient for e-money. International usage of the system is possible if the encryption is not made "general purpose," but is rather restricted to the use inside the user's software.

# 3 Processes and Protocols

We first enumerate the basic processes taking place in *VarietyCash*, and the parties involved in them. Later we describe in detail the requirements and protocols that realize them for the two more relevant

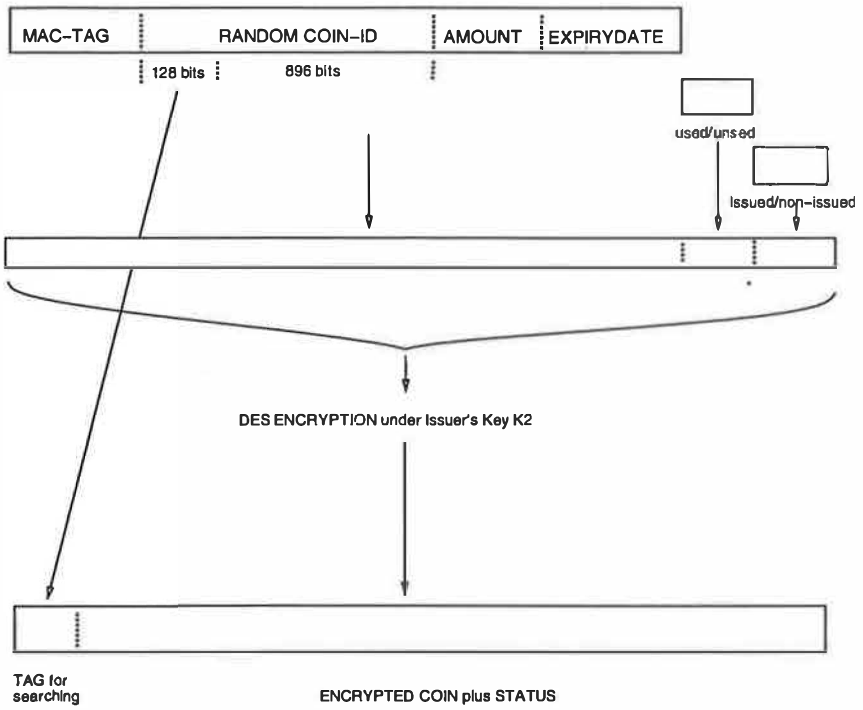


Figure 3: *Encrypted coin in the Issuer's coin database.*

processes: *Coin Purchase* and *Payment.*

- Registration – A party chooses an identity, and has his/her public key registered at the issuer.
- Enrollment ($PY, I$ or $PA, I$) – An already registered participant enrolls for a role such as coin purchaser or redeemer.
- Coin Purchase ($CP, I, B$) – The withdrawer specifies what kinds of coins s/he wants, and a corresponding set of coins is then issued and sent to the withdrawer. The coins are paid for by funds of value equal to the total dollar value of the issued coins, which the withdrawer authorizes the issuer to get from his/her bank account.
- Payment ($PA, PY, I$) – The payer pays to the payee a requested sum, using one or more coins totalling to the requested value. The payee immediately (i.e., on-line) validates the coins with the issuer, and may either obtain new coins in return, redeem the coins, or aggregate them for later redemption.
- Change ($CH, I$) – The coin-holder gives the issuer a set of coins, and also specifies what kinds of coins he wants in change, the total dollar value of the requested coins being the same as that of the provided coins. Coins corresponding to the request are then issued to the coin-holder.
- Redeem ($RD, I, B$) – The redeemer gives some set of coins to the issuer, and the latter turns them into real money in the redeemer's bank account.
- Refresh ($Rf, I$) – The refresher turns in old (expired) coins for an equivalent value in new coins.
- Refund – A non-anonymous coin holder can request the issuer to resend him his coins in case of a failure (e.g., disk crash).

Again, we note the distinction between the Registration and Enrollment processes. Intuitively, Registration is a simpler, on-line process that will let a user participate in the transactions. It basically consists of chooshing and ID (possibly a *pseudonym*) and a secret/public key pair, and making sure that requirements such as availability and security (e.g., minimizing spoiler attacks) are met. On the other hand, Enrollment is accomplished by a combination of on-line and out-of-band steps. It comprises steps such as providing DDA/credit card information, validation of this information, and issuance of initial amount. Being enrolled allows a user to play an "active" role in the system, in the sense of generating money conversions from regular to electronic, and viceversa. We leave the details of these two important processes for the full version of the paper, and will assume in the following description that they have already taken place.

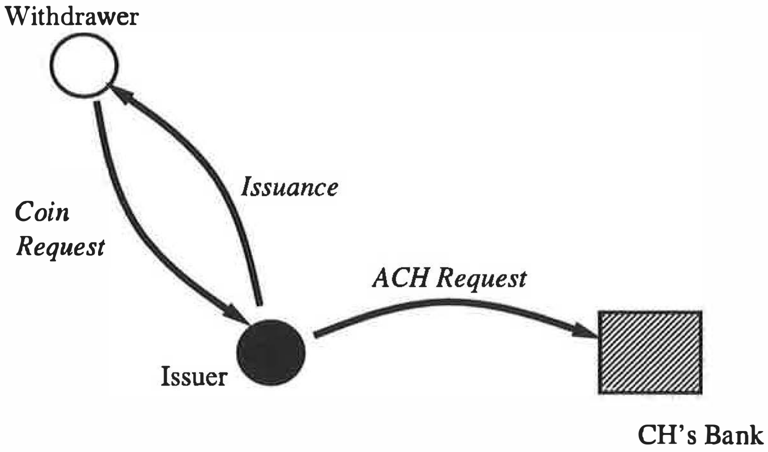


Figure 4: *The Coin Purchase process.*

### 3.1 Coin Purchase

The Coin Purchase process involves the coin purchaser, the issuer and the coin purchaser's bank. The coin purchaser specifies how much he wants in e-coins, and in what denominations, and provides the information to make the coin purchase from the bank. The issuer makes the coin purchase and then issues the coins.

The requirements are as follows:

**W1**– *Valid transactions go through.*

**W2**– *Can't get coins for nothing.* It is not possible to get coins without paying for them: if a party ends up getting a certain dollar value of valid coins, then the issuer has the corresponding funds from the same party's bank account.

**W3**– *Can't create false debits.* An adversary may want to play spoiler: it doesn't want coins, but wants the coin purchaser's account to be unnecessarily debited. This should not be possible. That is, it is not possible for an adversary to create a fake coin request which leads the issuer into debiting the coin purchaser's account.

A protocol for the coin purchase process is shown in Figure 5. The field W-DESC contains two things. First, the type and number of coins he wants: this is a list of denominations, and, for each denomination, the number of coins of that denomination that are desired (see Section 2.4). Second, information necessary to enable the issuer to get paid, in real funds of equal value to the total dollar value of the coins. Here we take the case that this payment is made by coin purchase from the coin purchaser's bank, so this information includes the bank name and address, and the coin purchaser's account number.[1]

Note that this protocol does not make use of a certification authority. It assumes that the parties have each other's public keys and certificates already cached. The coin purchaser has the issuer ID and public key in his purse from enrollment, and similarly the issuer has the coin purchaser ID and public key in his database from enrollment.

There are three transactions: the coins request, in which the coin purchaser asks for the coins and provides the bank information; the execution, in which the ACH transaction is done; and the final issuance of coins. The protocol is designed to guarantee both privacy and authenticity of the data. This is to protect the coin purchase information and the coins that are issued. It must also provide freshness. For efficiency's sake we use a key exchange protocol to get a session key $K$ under which later messages are encrypted or MACed. However, the coin purchase information is digitally signed for non-repudiability. We now go over the transactions in more detail.

(1) Coin Request. The coin purchaser requests that a certain amount in coins be returned to him, and authorizes the issuer to withdraw this amount from his bank account. The protocol begins with a key exchange which issues the key $K = K_{CP} \oplus K_I$ to both parties:

(1.1) WRequest1. The coin purchaser chooses a random number $K_{CP}$, and then encrypts

[1] Another possibility is that this payment is made by credit card, in which case an *i*KP/SET-type protocol [3, 21] may be used, instead of the protocol we are describing here. The issuer would play the role of the merchant in SET.

- **Fields:**

| | |
|---|---|
| W-DESC | *Coin Purchase description–* amount, denominations of desired coins, information and authorization required to make debit at bank. |
| $R_X$ | Random challenge chosen by party $X$ |
| $K_X$ | Random number chosen by party $X$ |
| $K$ | $K_{CP} \oplus K_I$ |

- **Protocol Flows:**

$$\text{WRequest1} \;:\; CP \xrightarrow{\mathbf{E}^{\bullet}_{I}(\mathrm{ID}_{CP}, K_{CP})} I$$

$$\text{WRequest2} \;:\; CP \xleftarrow{\mathbf{E}^{\bullet}_{CP}(\mathrm{ID}_{I}, K_{I}, R_{I})} I$$

$$\text{WRequest3} \;:\; CP \xrightarrow{\mathbf{e}_K(\text{W-DESC}, \ \mathbf{S}_{CP}(\mathrm{ID}_I, R_I, \text{W-DESC}))} I$$

$$\text{WExecution} \;:\; I \xrightarrow{\text{ACH-Req}} B$$

$$\text{Issuance1} \;:\; CP \xleftarrow{\overbrace{\mathbf{e}_K(\mathrm{Coin}_1, \ldots, \mathrm{Coin}_n)}^{\text{EncC}}, \ \mathbf{mac}_K(\mathrm{ID}_I, \mathrm{EncC})} I$$

$$\text{Issuance2} \;:\; CP \xrightarrow{\mathbf{mac}_K(\mathrm{ID}_{CP}, R_I)} I$$

Figure 5: *Coin Purchase rotocol*

his identity $\mathrm{ID}_{CP}$ and $K_{CP}$ under the public encryption key of the issuer, using the plaintext aware encryption algorithm. The ciphertext is passed to the issuer.

(1.2) WRequest2. The issuer applies the plaintext-aware decryption algorithm to the received ciphertext. If this algorithm rejects the text as non-authentic then he rejects; else be obtains and records the identity $\mathrm{ID}_{CP}$ of the withdrawer and $K_{CP}$. Now he chooses a random number $K_I$ and also a random nonce $R_I$. He uses $\mathrm{ID}_{CP}$ to retrieve $\mathrm{PK}_{CP}$ and then encrypts $\mathrm{ID}_I, K_I, R_I$ under $\mathrm{PK}_{CP}$ using the plaintext aware encryption algorithm. The ciphertext is sent to the coin purchaser. The value $K = K_I \oplus K_{CP}$ is stored as the session key.

(1.3) WRequest3. The withdrwawer applies the plaintext-aware decryption algorithm to the received ciphertext. If this algorithm rejects the text as non-authentic then he rejects; else be obtains and records the identity $\mathrm{ID}_I$, and the numbers $K_I, R_I$. He checks that the identity is really that of the issuer by matching it with the value in his purse. He forms the session key $K = K_{CP} \oplus K_I$. Now he forms the indicated flow, which contains W-DESC and a signature, the whole encrypted under the shared session key $K$ to ensure privacy of the bank coin purchase information. Note the nonce $R_I$ is included in the signature to ensure freshness.

(2) Execution. The issuer now uses the coin purchase information and authorization provided by the customer to make the ACH transaction of coin purchase from the bank.

(2.1) WExecution. The coin purchaser uses $K$ to decrypt the ciphertext and obtain W-DESC and the signature. He checks that the signature is valid, and stores it. Now he uses the information in W-DESC to make the ACH request. The issuer then waits a suitable period (which can range up to the order of days). If there is anything wrong, the bank sends a reject within this period; else the funds are in the issuer account. Now the issuer is ready to issue the coins.

(3) Issuance. The issuer forms e-coins $\text{Coin}_1, \ldots, \text{Coin}_n$ of the denominations requested in W-DESC. (These coins may have been created earlier and are archived, or may be formed at this time.) Then:

(3.1) Issuance1. The issuer encrypts the coins under the session key $K$ to get EncC. This ciphertext is then authenticated, also under $K$, by computing $\mathbf{mac}_K(\text{ID}_I, \text{EncC})$. The ciphertext and the MAC are sent to the coin purchaser.

(3.2) Issuance2. The coin purchaser checks that the MAC is correct. (This means the coins are really from the issuer.) Then he decrypts the ciphertext to get the coins, which go into the purse. He now issues a final acknowledgment, consisting of the issuer nonce $R_I$ MACed under the session key.

In the full paper we show how requirements **W1–3** are met. We now turn to the description of the Payment process.

## 3.2 Payment

The Payment process involves the payer (e.g., a customer), the payee (e.g., a merchant) and the issuer. The requirements are as follows:

**P1–** *Valid payments go through.* If the payer transfers a certain amount in valid coins, and if these coins are as yet unspent, then, after checking with the issuer, the payee accepts the payment. (He may end up with refreshed coins, or have redeemed them, or aggregated, as he wishes.)

**P2–** *Accepted payments are valid.* If after checking with the issuer a payee accepts a payment, then he knows that the refreshed coins he has obtained are valid. In particular, already-spent coins are detected: If a payer uses an already-spent coin then (by checking with the issuer) the payee will detect it, and the payee will not accept the payment.

**P3–** *Payment is for the goods or services the parties have agreed on.* An adversary $A$ cannot divert a payment by the payer to $A$'s advantage, or even change the order description in "spoiler" ways. This is an optional requirement, which can be provided given an external certification authority.

**P4–** *Payer is informed of double spending.* In case the issuer detects double spending, the payer should be told his coins are bad, and be sure that the issuer thinks so.

We provide two basic kinds of payment protocols: payment with refresh (i.e., the payee obtains new coins) and payment with redemption (the payee is enrolled, and obtains real funds). In this abstract we only describe the former; payment with redemption has the same flavor. The same applies to payment with *aggregation.* The flows involved in payment with refresh are shown in Figure 6, and the protocol in Figure 7. The V-DESC field indicates which option is being used. In addition, it includes whatever information is needed for the option being used. For example, if it is refresh, the V-DESC field includes the type and number of the desired coins; if it is redemption, the V-DESC field includes the bank name, address and the account number.

The payment protocols have certain optional flows. They are indicated in square brackets, for example $[\mathbf{S}_{PY}(\mathcal{H}(\text{Com}))]$ means providing this signature in the first flow is an option. The issue here is certificates. The basic protocol does not need a certification authority: it is enough that the issuer have the public keys of the participants. But for the extra functionality of order protection and receipt, an external certification authority is needed to provide the payer with the public key of the payee. We now describe how the flows are computed.

(1) Invoice. This transaction consists of a single flow in which the payee provides the transaction ID. The latter is a randomly chosen number which uniquely identifies the transaction. For confirmation of amount and order information, it is suggested that this be accompanied by a signature of the common information.

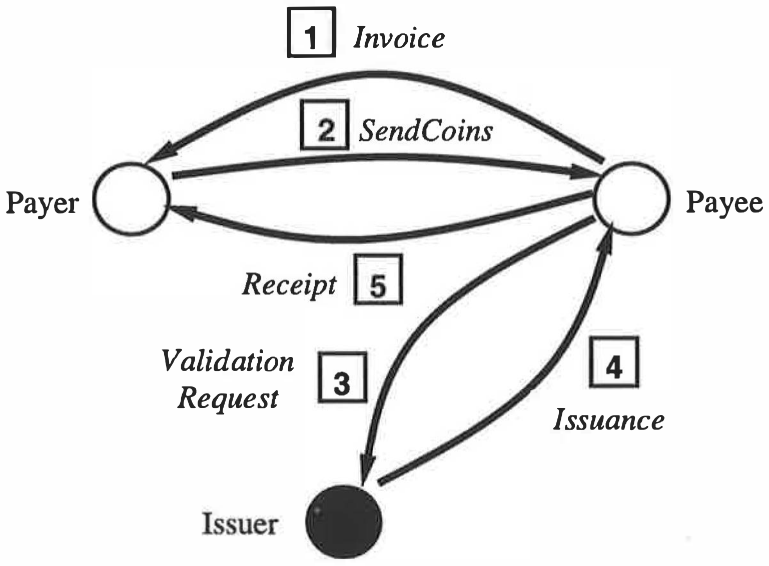


Figure 6: *Payment with Refresh: Flows*

(2) SendCoins. The payer picks from his purse a collection of coins $\text{Coin}_1, \ldots, \text{Coin}_n$ whose total dollar value equals the amount to be paid. (If the purse happens to not currently hold this amount, but holds coins of total dollar value which is larger, the payer can go through a change transaction to get change, and then resume the payment. If the purse has insufficient funds, the payer will have to make a coin purchase, and, since this is a lengthy process, he will probably stop the payment here and re-start when he has the funds.) The coins are put in an envelope by encrypting them (and the identity of the payee) under the public key of the *issuer*. The ciphertext is transmitted to the payee.

(3) Validation Request. The payee cannot open the envelope; he never sees the coins. Instead, he forwards them to the issuer for validation, along with V-DESC which indicates whether he wants refresh, redemption or aggregation. The payee also includes in V-DESC the amount, to guard against the payer paying less than the agreed amount. This is done, for privacy, under cover of an encryption under the issuer's public key. Also in the scope of the encryption go the payer identity, the transaction id, and a number $K_{PY}$, chosen at random, which will be used to derive a session key.

At this point the process is different depending on whether we are doing refresh or redeem. We continue to describe the refresh case.

(3) Issuance. The issuer decrypts the ciphertext to obtain $\text{ID}_{PY}, K_{PY}, \text{TID}_{PY}, \text{V-DESC}, \text{EncC}_1$. He then decrypts $\text{EncC}_1$ to get the coins which were sent by the payee. The validity of these coins is checked, as also the fact that the total value of these coins matches the amount claimed by the payee that is present in V-DESC. Now new coins are issued, of the type specified in V-DESC, via two flows:

(3.1) Issuance1. The issuer picks a number $K_I$ at random and forms the session key $K = K_I \oplus K_{PY}$. The session key, together with $\text{ID}_I$, are encrypted under the public key of the payee, and the resulting ciphertext is transfered to the payee. Also the issuer encryptes the new coins under $K$; then MACs this ciphertext and some other stuff as shown. The second ciphertext and the MAC are also sent to the payee.

(3.2) Issuance2. The payee acknowledge having received the new coins by sending a message signed under the session key $K$.

(4) Receipt. The last flow is optional, and consists of a receipt, from payee to payer, that the payer's payment was accepted by the issuer.

We have omitted from the protocol picture the flows related to error conditions, such as the issuer informing the payer if his coins are bad, or the issuer informing both parties if the claimed and paid amounts do not match. The issuer would sign the bad coins and the error statement, and pass this to the payee, who in turn passes it to the payer.

Several spoiling attacks are possible. For example an attacker could flip some bits in the MAC in the Issuance1 flow, making the payee reject. In a seemingly more sophisticated attack, he can remove the ValidationReq flow sent by the payee and substitute a fake one which contains the same information except that the value of $K_{PY}$ is different. (Note he is in possession of all information except $K_{PY}$ so can indeed do this.) Then the payee will again reject the Issuance1 flow since he will recover the wrong session key. However, such attacks do not really help the attacker. These kinds of spoiling attacks are unpreventable, and handled by appropriate error handling and re-transmission.

# 4 Integrating Card Cash

## 4.1 Card-based systems

Typically, a card-based e-money system is an electronic payment system based on a tamper-proof device, with the properties of being pseudo-anonymous, offline, and non-circulating. The term non-circulating refers to the fact that the monetary value once paid cannot be reused offline (by the payee of the first transaction) and must be redeemed. Being an offline system, it requires tamper-proof smartcards holding monetary values (for making payments), as well as tamper-proof devices for accepting payments. The devices for accepting payments require even higher level of protection as they may contain certain global cryptographic keys.

The e-money system outlined in Sections 2 and 3 is a network-based, online payment system. In this section we show how that system can be combined with this card-based system to offer additional services and functionality, i.e., *VarietyCash*. For example, the combined system allows transfer of monetary value from one system to another and vice versa. The advantages of one system can then be used in the other. Fortunately, these transfer protocols can be built at a high level, on top of the two systems.

## 4.2 Cardcash

We assume the card-based system has the following components:

- Purse smartcard (smartcard for short) (*SM*),
- load/unload device,
- load/unload server (*LUS*),
- purchase device,
- purchase SAM (Security Access Module), and
- collection server.

In addition, there maybe one or more payment and clearing agencies (*AG*) which are not part of the system, but are needed for transfering the real money to electronic money and vice versa.

The cryptographic protocols used in these systems typically use symmetric keys (e.g., DES). The security is hierarchical in nature. In other words, there are devices which have the capability of generating their own keys, whereas devices higher up in the hierarchy can generate keys of devices immediately lower in rank using identification information of the lower ranked device. For example, a smartcard is lower ranked than a Purchase Security Access Module(PSAM). The PSAM has a global key $K$, whereas the purse just has a derived key $f(K, ID)$, where $f$ is a predefined one-way function, and ID is the identification number of the smartcard. The PSAM can dynamically generate the derived key once it has the identification of the smartcard.

In general, the keys are "segmented" for further security. In such a scheme there maybe many global keys for a particular device class. In this abstract we will simplify the presentation by assuming no segmentation of keys.

We now turn to the description of some of the "native" card-based system protocols, and the transfer protocol between systems.

## 4.3 Load protocol

There are many modes of loading value onto the card, such as off-line load using Load Security Access Modules (LSAM), on-line account-based loads, and on-line non-account-based loads. In this abstract we will only cover on-line account-based loads. In the on-line account-based load, there is a payment agency involved with which the card holder has an account (or registration). The load is mediated by the payment agency, and the account is debited.

Informally, when a user with a card wants to load value onto his card (using an on-line account based method), he communicates with his payment agency using the load/unload device. Once a load of a particular value is requested by the card, the payment agency (which acts as a trusted party), instructs the load/unload server to load the monetary value onto the card (while guaranteeing payment). All further messages between the load/unload server and card go via the payment agency (or the client), and the payment agency and the load/unload server commit the transaction based on acknowledgments from the two end parties. The load/unload server logs the transaction, and clears the payment with the payment agency via a clearing system.

A sketch of the protocol is shown in Figure 8. There are three parties invloved: the load/unload server (*LUS*), the card (*SM*), and the payment agency (*AG*). The first step of the Load protocol (not shown) is itself a high level protocol between an account holder and a payment agency. This protocol ensures a monetary value transfer from the smartcard holder to the payment agency. For example, if the payment agency is your bank, this payment protocol is an electronic authorization to debit your account. As to how this protocol is defined is independent of the card-based system, except for the fact that it should be able to associate a *SM* ID, a unique transaction ID for this *SM* ID, and an amount with this transaction to be used in the composite transaction.

All the messages between *SM* and *LUS* go via the payment agency (or the load/unload client in the payment agency). However, note that in this protocol there is no way for the server *LUS* to prove to the payment agency that the load into the card actually happened, since the key is symmetric. In other words the signature (i.e., signed ack) in the protocol—it could have been generated by the server itself. Thus, the server is assumed to be a trusted party.

The Unload protocol is symmetrically opposite to the one of Figure 8.

### 4.4 Transfer protocols and complex payment transactions

We now briefly describe the transfer protocols between card- and software-based systems. Effectively, this is a composition of an online protocol between a user and a load/unload server, followed by an online protocol between the same user and the issuer of Section 2. Specifically, an account-based Unload protocol followed by a Coin Purchase protocol similar to that of Figure 5. The issuer acts as the payment agency in the Unload protocol. Call this combined protocol Cardcash-to-Softcash.

The reverse direction, software cash to card cash, is composed of the Redeem protocol between the user and issuer, followed by the Load protocol of Figure 8 between the same user and a load/unload server. Again, the issuer acts as the payment agency in of the Load protocol. Call this protocol Softcash-to-Cardcash.

In the same way, transactions between users from the same or different payment "media" are also enabled. For example, the payment from one card user to another is achieved as follows. This type of transfer can be accomplished by any intermediary, in particular, the issuer of Section 2:

1. The payer runs Cardcash-to-Softcash;
2. the payer, payee and issuer run the Payment with Redeem protocol (Section 3);
3. the payee runs Softcash-to-Cardcash.

Note that in the card-based system a clearing system is used for transfering money back and forth between the load/unload servers and the payment agencies. Thus, there is not much of an overhead in running these composed protocols. At the end of the three steps above the payment agency (the issuer) and the load/unload server are even.

Similarly, card payers can pay software payees with not much of an overhead, since a clearing system is involved. Here, the unload server owes money to the issuer. In the reverse direction, the issuer owes money to the load server.

## Acknowledgements

We are thankful to the anonymous referees for the many comments regarding comparison with other payment systems, as well as spoiling attacks against some of the protocols.

## References

[1] M. Abdalla, M. Bellare and P. Rogaway. DHAES: An encryption scheme based on the Diffie-Hellman problem. Manuscript.

[2] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway. Relations among notions of security for public-key encryption schemes. *Advances in Cryptology – Crypto 98 Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.

[3] M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik and M. Waidner. iKP – A Family of Secure Electronic Payment Protocols. *Proc. First Usenix Workshop on Electronic Commerce*, pp. 89-106, New York, June 1995.

[4] M. Bellare, J. Garay and T. Rabin. Fast Batch Verification for Modular Exponentiation and Digital Signatures. *Advances in Cryptology – Eurocrypt 98 Proceedings*, Lecture

Notes in Computer Science Vol. 1403, K. Nyberg ed., Springer-Verlag, 1998.

[5] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption. *Advances in Cryptology – Eurocrypt* 94 *Proceedings*, Lecture Notes in Computer Science Vol. 950, A. De Santis ed., Springer-Verlag, 1994.

[6] D. Bleichenbacher. A chosen ciphertext attack against protocols based on the RSA encryption standard PKCS #1. *Advances in Cryptology – Crypto* 98 *Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.

[7] J.-P. Boly *et al.* The ESPRIT Project CAFE - High Security Digital Payment Systems. *ESORICS '94*, LNCS 875, Springer-Verlag, Berlin 1994, 217-230. <http://www.informatik.uni-hildesheim.de/FB4/Projekte/sirene/projects/cafe/index.html>.

[8] D. Chaum. Blind signatures for untraceable payments. *Advances in Cryptology - Crypto 82 Proceedings*, D. Chaum, R. Rivest & A. Sherman eds., Plenum Press, New York, pp. 199-203.

[9] CyberCash. The CyberCash(tm) System - How it Works. <http://www.cybercash.com/cybercash/cyber2.html>.

[10] DigiCash. About ecash. <http://www.digicash.com/ecash/ecash-home.html>.

[11] D. Dolev, C. Dwork, and M. Naor, Non-malleable cryptography. *Proceedings of the* 23rd *Annual Symposium on the Theory of Computing*, ACM, 1991.

[12] W3C Joint Electronic Payments Initiative (JEPI). <http://www13.w3.org/ECommerce/Overview-JEPI.html>.

[13] M. Jakobbson and M. Yung. Revokable and versatile electronic money. *Proceedings of the Third Annual Conference on Computer and Communications Security*, ACM, 1996.

[14] Mondex. <http://www.mondex.com>.

[15] G. Medvinsky. NetCash: A framework for electronic currency. Ph. D thesis, Dept. of Computer Science, USC, 1997.

[16] G. Medvinsky, C. Neuman. NetCash: A design for practical electronic currency on the Internet. *Proceedings of the First Annual Conference on Computer and Communications Security*, ACM, 1993. <ftp://prospero.isi.edu/pub/papers/security/netcheque-requirements-compcon95.ps.Z>.

[17] C. Neuman. Proxy-based authorization and accounting for distributed systems. *Proc. 13th International Conf. on Distributed Computing Systems*, pp. 283-291, May 1993.

[18] C. Neuman, G. Medvinsky. Requirements for Network Payment: The NetCheque Perspective. *IEEE COMPCON*, March 95. <ftp://prospero.isi.edu/pub/papers/security/netcheque-requirements-compcon95.ps.Z>.

[19] M. Sirbu, J. D. Tygar. NetBill: An Internet Commerce System. *IEEE COMPCON, March 95.* <http://www.ini.cmu.edu/netbill/CompCon.html>.

[20] Secure Electronic Marketplace for Europe (SEMPER). <http://www.semper.org/>.

[21] SET. <http://www.mastercard.com/set/>.

[22] D. Tygar. Atomicity in electronic commerce. Invited talk and paper, *Proc. Fifteenth Annual ACM Symposium on Principles of Distributed Computing,*, pp. 8-26, Philadelphia, May 1996.

[23] M. Wegman and L. Carter. New hash functions and their use in authentication and set equality. *J. of Computer and System Sciences* 22, pp. 265–279, 1981.

- **Fields:**

| | |
|---|---|
| P-DESC | *Purchase description*– amount, description of goods, payment method or mechanism. Assumed part of starting information of payer and payee. |
| $\mathrm{TID}_{PY}$ | *Transaction ID*– a number generated by the payee which is uniquely associated to this transaction |
| Com | P-DESC, $\mathrm{TID}_{PY}, \mathrm{ID}_{PA}, \mathrm{ID}_{PY}, \mathrm{ID}_{I}$ |
| V-DESC | Verification and execution request text. Indicates one of three options (refresh, immediate redeem, or aggregate) and provides corresponding data. Includes amount. |
| $K_X$ | Random number chosen by party $X$ |
| $K$ | $K_{PY} \oplus K_I$ |

- **Protocol Flows:**

Invoice : $PA \xleftarrow{\mathrm{ID}_{PY},\ \mathrm{TID}_{PY},\ [\mathbf{S}_{PY}(\mathcal{H}(\mathrm{Com}))]} PY$

SendCoins : $PA \xrightarrow{\overbrace{\mathbf{E}^*_I(PY, \mathrm{TID}_{PY}, \mathrm{Coin}_1, \ldots, \mathrm{Coin}_n)}^{\mathbf{EncC_1}}} PY$

ValidationReq : $PY \xrightarrow{\mathbf{E}^*_I(\mathrm{ID}_{PY},\ K_{PY},\ \mathrm{TID}_{PY},\ \text{V-DESC},\ \mathrm{EncC}_1)} I$

Issuance1 : $PY \xleftarrow{\mathbf{E}^*_{PY}(\mathrm{ID}_I, K_I),\ \overbrace{\mathbf{e}_K(\mathrm{Coin}'_1, \ldots, \mathrm{Coin}'_{n'})}^{\mathbf{EncC_2}},\ \mathbf{mac}_K(\mathrm{ID}_I, \mathrm{TID}_{PY}, \mathrm{EncC}_2)} I$

Issuance2 : $PY \xrightarrow{\mathbf{mac}_K(\mathrm{ID}_{PY}, \mathrm{TID}_{PY})} I$

Receipt : $PY \xrightarrow{[\mathbf{S}_{PY}(\mathrm{ID}_{PA}, \mathcal{H}(\mathrm{Com}), \mathrm{EncC}_1)]} PA$

Figure 7: *Payment with Refresh protocol.*

- **Fields:**

| | |
|---|---|
| $SM_{ID}$ | Smartcard's unique ID |
| $TID$ | A unique transaction ID for the smartcard |
| $Amount$ | Monetary value to be loaded |
| $\mathbf{e}_{Load-key(SM_{ID})}$ | Encrypted under the derived Load key of the $SM$, stored as it is on the $SM$ , and computable by the load server using its own global key and $SM_{ID}$ |
| $R_X$ | Random challenge chosen by party $X$ |
| $\mathcal{H}$ | A standard one-way hash function |

- **Protocol Flows:**

LInitialize1 : $LUS \xleftarrow{TID, SM_{ID}, Amount} AG$

LInitialize2 : $SM \xleftarrow{\mathbf{e}_{Load-key(SM_{ID})}(TID, R_{LUS}), TID} LUS$

LDebit : $SM \xrightarrow{\mathbf{e}_{Load-key(SM_{ID})}(\mathcal{H}(TID, R_{LUS}), R_{SM}), TID} LUS$

LCredit : $SM \xleftarrow{\mathbf{e}_{Load-key(SM_{ID})}(\mathcal{H}(R_{SM}), Amount), TID} LUS$

LAck1 : $SM \xrightarrow{\mathbf{e}_{Load-key(SM_{ID})}(\mathcal{H}(TID, R_{SM} \oplus R_{LUS})), TID} LUS$

LAck2 : $LUS \xrightarrow{TID, \text{``}OK\text{''}} AG$

Figure 8: *Load protocol*

# NetCents: A Lightweight Protocol for Secure Micropayments

Tomi Poutanen
University of Toronto
poutanen@embanet.com

Heather Hinton
Ryerson University
hhinton@ee.ryerson.ca
Toronto, Canada

Michael Stumm
University of Toronto
stumm@eecg.toronto.edu

## Abstract

NetCents is a lightweight, flexible and secure protocol for electronic commerce over the Internet that is designed to support purchases ranging in value from a fraction of a penny and up. NetCents differs from previous protocols in several respects: NetCents uses (vendor-independent) *floating scrips* as signed containers of electronic currency, passed from vendor to vendor. This allows NetCents to incorporate decentralized verification of electronic currency at a vendor's server with offline payment capture. Customer trust is not required within this protocol and a probabilistic verification scheme is used to effectively limit vendor fraud. An online arbiter is implemented that will ensure proper delivery of purchased goods and that can settle most customer/vendor disputes. NetCents can be extended to support fully anonymous payments. In this paper we describe the NetCents protocol and present experimental results of a prototype implementation.

## 1 Introduction

This paper introduces NetCents, a novel protocol for electronic commerce transactions. NetCents was designed to support micropayment transactions, payments that range in value from a fraction of a penny to several dollars, and was extended to handle larger value purchases. A number of systems capable of supporting micropayments have been proposed and implemented, but none have established themselves as a de facto standard. The lack of a standard payment protocol that handles the full range of payments has arguably limited the growth of consumer driven electronic commerce on the Internet. Electronic payment protocols in general require a tradeoff between transaction security and transaction cost. NetCents effectively bridges that gap and satisfies the requirements of a universal payment mechanism. The work builds on the *NetBill* [CTS95], *DigiCash* [Ch95] and *Agora* [GaSi96] protocols and in particular the *Millicent* [Man95] protocol.

The key innovation of NetCents is its use of *floating scrips*. A NetCents floating scrip is a signed container of electronic currency passed from one vendor to another, such that it is active at only one vendor at a time. A scrip has a monetary value or balance associated with it, as well as a unique public and private key. Public key cryptography is used for increased security and privacy. It also enables NetCents to provide the anonymity of cash, yet the buyer can dispute a purchase effectively to an online arbiter.

NetCents functions in an offline fashion, without the need to contact a verification server in the common case. Floating scrips are vendor-independent and are passed from vendor to vendor without the need for central authorization. NetCents is scalable and supports multiple currencies and issuing authorities (or mints) and it provides adequate security, privacy and non-repudiation.

NetCents was designed to minimize operational costs. The bank is only required for offline batch processing and customer services. Vendor overhead is carefully minimized by offloading the majority of the computation load onto the purchaser; in the common case, payment verification is reduced to two modular multiplications. Vendor communication costs are reduced by distributing the scrips in a LRU fashion, in an attempt to exploit the users' tendency to shop repeatedly at the same locations

In the following section we discuss some current electronic commerce protocols. The third section describes the NetCents payment protocol in detail. We then discuss the security properties of the protocol in section four. Section five describes our implementation of the NetCents. Finally, we compare NetCents with other micropayment protocols.

## 2 Related Work

### 2.1 Basic Protocol Properties

In a computerized transaction system, a transaction should have four characteristics: atomicity, consistency, isolation and durability [GR93], which are commonly referred to as the ACID properties. The ACID properties have recently also been used to evaluate electronic payment systems [CTS95, CHTY96]. The atomicity property has been further subdivided in the context of electronic commerce to include the delivery of purchased goods as part of the evaluation of a payment transaction [Ty96]. This is especially relevant in the realm of Internet commerce, where communication channels are not reliable.

**Money atomicity** refers to the atomic transfer of money, where a transaction either fully completes or it does not complete at all and does not create or destroy money. A transaction is **goods atomic** if it is money atomic and if the customer will receive the goods purchased if and only if the merchant is paid. A goods atomic transaction provides an atomic swap of electronic goods for funds. If a protocol is goods atomic and allows a consumer and merchant to prove exactly what was delivered, it satisfies **certified delivery** requirements. In case of a dispute, this evidence can be shown to a trusted arbiter to prove exactly what was delivered.

Other implementation and usability issues in Internet payment systems include anonymity, scalability, divisibility, transferability, interoperability, and non-repudiation. Anonymity implies that the identity of the buyer is not revealed in the course of a transaction. Most protocols achieve anonymity against vendors and snoopers (partial anonymity) but not against the buyer's bank (full anonymity) which can track the user's shopping habits. System scalability is of paramount importance in order to accommodate a worldwide user base without bottlenecks and single points of failure. Divisibility of currency is desired in order to pay amounts of arbitrary value. Transferability is the ability to transfer funds to other users or financial institutions. Interoperability requires the system to function with multiple financial institutions and in many currencies. Finally, non-repudiation provides proof of integrity and origin of an online transactions that can be verified by any party – a requirement for effectively policing online fraud.

## 2.2 Hybrid Payment Systems

Currently, most online purchases involve the entry of credit card information over a secure connection. However, there are security concerns in the form of the delivery and storage of credit card information, and the existence of Trojan sites [TW96] that mimic legitimate vendors in an attempt to gain credit card numbers. Also, not only is manual entry cumbersome but there is also a high minimum transaction charge. This charge can be amortized over several payments at one vendor by using accounts. Account-based payments systems have further problems: the cumbersome registration process discourages "shopping around". There is no policing of promised service from the online service, and the customer may have a difficult time withdrawing the balance in her account.

## 2.3 Online Payment Protocols

With an online protocol, a central payment authority is contacted to authorize each transaction. In general, online systems (if designed and implemented properly) secure the merchant and the bank against customer fraud, since every payment is approved by the customer's bank. Customers, however, may counter theft or loss of their electronic money, and they may be cheated by merchants via misrepresentation of goods or failed delivery. The primary disadvantage of online authorization is the associated per transaction cost, imposed by the requirement for a highly reliable and efficient clearing system at the customer's bank.

Notable online protocols include CyberCoin (http://www.cybercash.com), DigiCash [Ch95], NetBill [CTS95], and, SET [SET97]. With CyberCoin, a client makes a payment by signing a fund transfer request to the merchant. The merchant submits this signed request to the bank for authorization of payment. Depending on the availability of funds in the buyer's account, the bank will reply to the merchant with a signed authorization or refusal. The scalability of the CyberCoin protocol is questionable since it relies on the availability of a single online bank. The protocol is not fully anonymous in that it allows the issuing bank to track every purchase. Finally, CyberCoin is not inexpensive: the system restricts payments to multiples of 25 cents, and charges a minimum authorization fee of 8 cents.

NetBill extends the above payment mechanism by supporting goods atomicity and certified delivery. This is accomplished with the use of encrypted delivery and signed price quotes and sales agreements, which can later be used by an arbiter to resolve disputes. The drawback of the protocol is the addition of extra messages, and the significant increase in the amount of encryption used.

DigiCash uses blind signatures to provide a fully anonymous coin-based payment system [Ch82]. Unfortunately, the need for sophisticated cryptographic functions for each coin requires added computational resources for the bank to validate the purchase. Flaws in the DigiCash protocol have been shown to lead to an inconsistent state violating the money atomic definition [CST95]. If transfer of DigiCash tokens from customer to merchant is interrupted, then it is possible that both or neither party may believe that it has legitimate access to the tokens. The protocol is also not goods-atomic, nor do the messages support non-repudiation.

Easily the most sophisticated protocol is the SET protocol, which was designed to facilitate credit card type transactions over the Internet. SET is secure, scalable and robust. The protocol is based on the RSA encryption protocol and utilizes 1024 bit public keys, and a 2048 bit root key. The protocol makes extensive use of cryptographic primitives that include hashes, public and private key encryption, digital signatures, dual signatures and certificates. The trust relationship is hierarchical, stemming from one root key and supported through digital certificates. This enables multiple certification authorities, issuing authorities (buyer banks), and acquirers (merchant payment agencies), facilitating wide scalability. Much of this security hierarchy is also preserved in the NetCents protocol.

SET security comes at considerable computation and communication cost. In particular, the system relies on traditional *interchange networks* when clearing payments between acquirers and issuers. Currently, interchange network services charge between 5 to 10 cents for timely and secure message passing between acquirers and issuers, whether it be a credit card or a debit card transaction [GS97]. This means that SET does not scale to the micropayment range. SET, unlike other simpler online protocols, does not offer full anonymity, non-repudiation or certified delivery of purchased goods.

## 2.4 Offline Payment Protocols

In an offline protocol, the merchant verifies the payment using cryptographic techniques, and commits the payment to the payment authority later, in an offline batch process. Offline systems were designed to lower the cost of transactions by delaying the clearing to a batch process. Offline systems, however, suffer from the potential of double spending, whereby the electronic currency is duplicated and spent repeatedly. Thus, offline protocols concentrate on detecting and limiting fraud, and in catching the fraudulent party. They are generally suitable only for low value transactions where accountability after the fact is sufficient to deter abuse.

Two notable offline protocols include Agora [GaSi96], Mini-Pay [HeYo96]. Agora allows its payment protocol to piggyback on existing HTTP Get-Request messages without additional communication overhead. Agora also adds non-repudiation and an online arbiter that functions much like NetCents'. Agora's method of fraud control by means of customer revocation messages may not scale to a worldwide reach, however. Mini-Pay is similar to Agora but does not use revocation messages. Instead, Mini-Pay requires a daily signed authorization from the customer's bank with a defined credit limit. As such, Mini-Pay is vulnerable to double spending up to the credit limit at multiple vendors. Other offline protocols include PayWord and MicroMint [RiSh96], Micro Payment Transfer Protocol (MPTP) [HalB95], and micro-*i*KP [HSW96]

## 2.5 Millicent

Digital Equipment's Millicent [DEC95, Man95] does not fall into either the online or the offline category, but rather is a distributed allocation of funds to merchants, who locally authorize payments. Grossly simplified, Millicent is an automated account based system. It introduces a scrip, which is a digital money that is honored by a single vendor. In contrast to NetCents, Millicent scrip is specific to only one vendor and the scrip must be cleared by the broker when relocating the scrip elsewhere.

A Millicent customer will have their electronic credit distributed at various site accounts on the Internet, with a common management interface - in effect, prepaying for access to a vendor, as in an account based scheme. On the first visit to an online merchant, the customer requests her broker (bank) for a signed scrip specific to the merchant. The scrip is transported to the merchant, who will subsequently authorize payments from the customer against that scrip. A scrip is analogous to a prepaid calling card, or a debit card specific to one merchant.

Once the scrip is fully used, the customer asks the broker to transmit additional (scrip) funds to the merchant. If the broker does not have sufficient available funds, then it can redeem the balance of a scrip from another vendor. Thus, while most small purchases can be made directly against scrip at the vendor, the broker is required to transfer funds between vendors. The protocol was designed on the assumption that online consumers, as in the real world, tend to shop repeatedly at the same merchants. If this assumption holds, then indeed the protocol is inexpensive to the broker due to its limited and mainly offline involvement. However, if a customer does not tend to revisit Internet merchants, then this protocol reverts to an online protocol – the broker is required in every transaction to shuffle funds from vendor to vendor. As with online payment protocols, this will result in longer payment latencies and a single point of failure. The movement of funds is especially troublesome when a customer balance nears zero and the number of scrips approaches one. At this point, scrip transfers become common, requiring broker involvement to redeem existing scrip and sign new ones.

Millicent does not use public key encryption, but uses shared keys with cryptographic hashes. Though this operation is substantially faster than public key cryptography, it implies that non-repudiation cannot be assured, and an online arbiter cannot be implemented. Shared keys also introduce communication overhead between the issuing authority and the account holder when creating and re-issuing scrips. Furthermore, the protocol is not fully anonymous, since the broker handles the customer's scrip transfers.

## 3 NetCents Protocol

The NetCents protocol was designed as a low cost, scalable electronic payment mechanism with the security properties of hard currency. It is based on the assumption that online shoppers tend to frequent the same vendor sites repeatedly on the Internet. With NetCents, as with Millicent, money is transferred in the form of scrip. A scrip consists of a public and a private portion. The public portion is called vendor scrip and consists of a short public key and a monetary balance. The vendor scrip is signed by the issuing authority (customer's bank) and distributed to vendors upon customer request. The private half of the scrip, the customer scrip, contains the corresponding private key and is concealed by the customer. Unlike Millicent, a NetCents scrip is not vendor-specific. This allows a customer to transfer a scrip between vendors directly without the involvement of a broker. Scrips are active at only one vendor at a time, thus enabling double-spending detection over a distributed network.

A scrip relocation algorithm minimizes scrip movement by drawing from the user's expected shopping behavior. Currently, a least recently used (LRU) policy is used in determining scrip movement. There is a fine balance between how much value a scrip can hold and how many scrips there are. A customer tending to spend all his money at only a handful of sites should have a small number of large valued scrips. A customer making many small payments to a broad range of vendors would benefit from a large number of low-value scrips. The allocation of scrips and scrip values can be fine-tuned over time by a customer software agent.

A purchase is executed with a signed electronic payment order (EPO). The EPO holds a snapshot of the scrip signed by the private key in the customer scrip. Successive payments are made against a scrip by presenting EPOs at declining balances. The EPO identifies the payer, payee, purchased item identifier, balance remaining in the scrip after the current transaction, and the time of the transaction. The EPO is encrypted within a single encryption block with the scrip private key.

The signing algorithm can be any one of a number of public key signature algorithms chosen to minimize the verification cost performed by the vendor and the bank. Our implementation uses the RSA public-key algorithm [RSA79]. By keeping the public key short, verification by the vendor and the bank is very fast [Sch96]. We chose a public key of 3, which reduces the EPO verification computation to two modular multiplications. The signing process performed by the customer is a more computationally expensive exponentiation. This mechanism effectively distributes the computational load of public key cryptography from the critical vendor server to the many buyers.

## 3.1 Protocol Participants

NetCents protocol participants include the Customer, Vendor, NetCents Root Certificate Server, Issuing Authority, Acquirer, Arbiter, and Blinding Site. Each participant is issued a digital certificate.

The **NetCents Root Certificate Server** periodically signs certificates for issuing authorities and acquirers and provides notification of revoked certificates. All participants have certificates that have been signed by either the root server or entities trusted by the root server.

An **issuing authority**, or issuer, is analogous to an online bank or mint that sells scrip, provides detailed transaction records, and guards against misuse and double spending. The issuing authority must possess a valid certificate signed with the NetCents root key.

An **acquirer** is an online financial institution that serves as a vendor's clearing center. The typical flow of money is from the customer to the issuing authority to the vendor's acquirer and finally to the vendor's bank account.

The **customer** creates a long term relationship with an issuing authority for the purchase of scrip. The issuing authority also provides a signed certificate that allows the customer to initiate future secure communications. A customer agent is the software interface that interacts with the system. A **vendor** is an online store that accepts NetCents scrip as a form of payment. A vendor has a long term relationship with an acquirer and holds a certificate signed by that acquirer.

**Arbiters** are mutually trusted, independent observers to transactions. Their function is to guarantee delivery of purchased goods and to provide for dispute resolution. **Blinding sites** are void, non-functional sites that are used to hide the previous location of a scrip.

Digital certificates are issued to every member in the NetCents community. All digital certificates have expiry dates, an IP address where applicable, and the member's (RSA) public key. The vendor certificate also includes a liability amount, used in vendor fraud detection. The customer certificate can optionally also include terms such as address, nationality, age or memberships that can be presented to vendors for proof of eligibility or discounts.

## 3.2 NetCents Transactions

In this section we outline the basic NetCents protocol. A more detailed description can be found in [Pou97].

The purchase of NetCents currency by a customer is performed via a more traditional and costly mechanism, such as an electronic transfer directly from the customer's bank to the issuing authority. The funds are released from the NetCents account in the form of scrips. The private customer scrip is transferred to the customer agent over a secure SSL connection.

The base NetCents purchase involves an exchange of messages between a customer and vendor. The transaction begins with the user requesting a quote for an item on a Web site (M0). The vendor returns the VendorID, ProductID, price quote, date and the vendor certificate, in a cleartext message (M1). The customer agent verifies the VendorID and vendor IP address against the vendor certificate. Once the vendor is verified, the customer agent will prompt the user to accept or decline the purchase from the vendor. If the customer has sufficient amount of money in a scrip at the vendor, then she will generate a signed EPO with the proper (post purchase) balance for transmission to the vendor (M2). The vendor verifies the ScripID, VendorID, Balance and Date in the EPO. If verified, the EPO will be stored for a later offline transaction with the bank, and the customer is supplied the goods and an acknowledgment (M3). As in Agora, the four messages can be piggybacked on current HTTP Get requests, with no additional messages needed for payment.

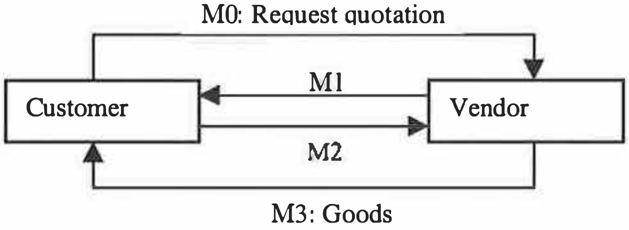


Figure 1: Basic NetCents purchase transaction

When a customer wishes to make a purchase at a vendor at which she does not have (enough) scrip, the customer agent instructs the vendor to fetch additional scrips, identifying the source – either another vendor or the issuing authority. The fetch request is accompanied with a signed EPO at its current balance identifying the new vendor. This transfer EPO serves two purposes. First, it provides the base balance of the scrip against which future payments are made. Second, the signed transfer EPO serves as proof to the vendor currently holding the scrip that the customer requested the transfer. This prevents fraudulent scrip transfer requests.

In an attempt to minimize scrip migration costs, the customer agent uses a least-recently used algorithm to select the vendor or issuer from which to request additional scrip. Prior to releasing the scrip to another vendor, the current vendor must first sign the scrip and its balance. This digital signature is used to guard against double spending initiated by a fraudulent vendor, as described later.

Some transactions may require a debit against more than one scrip. It is not sufficient to simply repeat the relocation and payment algorithm until enough scrip has been transferred to the merchant, as this can break the money atomic properties of the protocol. The vendor, acquirer, arbiter and issuing authority all require that a payment is one unbreakable unit. Hence, a multiple scrip payment procedure is as follows. Once all the scrips have been successfully transferred to the vendor, the post-purchase EPOs are sent in one atomic transaction. Atomicity is achieved by extending the EPOs to include a 128 bit MD5 hash of the concatenated scrip identifiers and ending balances. The individual signed EPOs are only useful if all signed EPOs can be presented to the issuing authority or an arbiter. This scheme is similar to SET's method of using dual signatures to link an order message with the payment instructions.

The distributed allocation of funds is efficient for micro-level payments, but becomes a burden for large value transactions. A large payment may require the system to fetch several scrips from other vendors in order to fulfill the payment. This translates to high payment latency and a higher probability of failure. Fortunately, intelligent fund allocation policies can overcome such inefficiencies. The issuer is advised to keep a sizeable balance available at the bank and only release an appropriate amount of the customer account as floating scrip for micropayments. For example, a customer that deposits $100 may receive only 10 one-dollar floating scrips and one $90 scrip that remains at the issuer. A large value transaction involves requesting the large value scrip from the bank, a payment against the scrip, and returning the scrip to the issuer – a process no more expensive than an online transaction. Thus, NetCents minimizes the cost of low value transactions by using floating scrips but it reverts to an online protocol for higher value transactions where per transaction costs are less of an issue to security.

Vendors collect their money from the issuing authority in offline batch processing at the end of the business day. For each scrip, the vendor presents to the issuing authority two signed EPOs at differing balances. In order to better model the banking process, the EPOs are first sent to the vendor's acquirer (clearing center) which then forwards them to the issuing authority. The highest and the lowest balance scrips are decrypted and verified. These may encompass multiple purchases by the customer at that vendor. The verification process is inexpensive at two modular multiplications per EPO. Once verified, the difference in balance represents the amount owing to the vendor, and the funds are transferred from the issuer to the vendor's account at the acquirer. The electronic funds transfer between the broker, issuing authority and the acquirer is handled independently from the NetCents protocol. The intermediary EPOs can be stored for transaction tracking, if desired.

Fund transfer between users is desirable but is complicated by the lack of trust and the lack of fraud control placed on users. The transfer of scrip must pass through an issuing authority in order to prevent double spending. This is the same as purchasing scrip at one issuing authority with someone else's scrip. The cost of involving a central server does impose a minimum per transaction cost.

## 3.3 Online Arbitration

In case of disputes, an online arbiter can use the signed EPOs in an audit. The product identifier within the EPO can be used to ensure delivery of a product to the buyer. The NetCents system requires that the customer can reload the purchased electronic goods by presenting the vendor with the EPO. That is, if M3 is lost and the goods are not delivered properly, then the customer can reinitiate the download by presenting the EPO. If the site fails to transmit the goods, then an online arbiter is contacted with the original EPO (M4 in Figure 2). The arbiter verifies the EPO, and contacts the vendor with the same EPO (M2'). If the vendor supplies the arbiter with the requested item, then the arbiter delivers the goods to the customer (M3'). However, if the arbiter is denied the goods, and the vendor refuses to roll back the transaction, then the issuing authority is notified (M5). The vendor is flagged, and the arbiter and customer are advised (M6) that the transaction has been cancelled. At the time of offline batch processing with the flagged vendor, the bank will request a full EPO history for the scrip. If the vendor attempts to use the discredited EPO, then the money is credited to the customer and not the vendor.

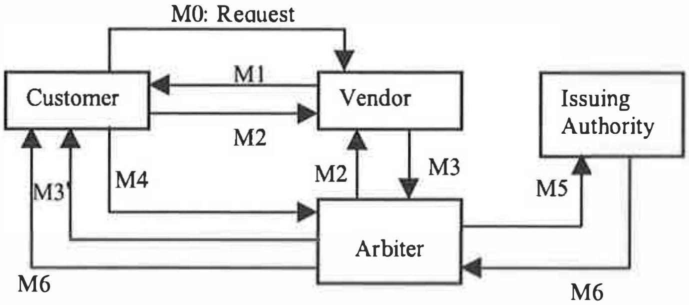


Figure 2: Online Arbiter Transactions

## 3.4 Anonymity and Privacy in NetCents

The protocol, as described, is not fully anonymous as the issuing authority can track the purchases against a scrip purchased by a certain client. An anonymous payment mechanism based on blind signatures has been developed as an "add on" and is described in length elsewhere [Pou97]. Like Chaum's DigiCash [Ch82], a customer can purchase scrip from an issuer by personally creating the scrip, digitally blinding it, and requesting the bank to sign it with a signature of an agreed value. When the customer is ready to make a purchase from a vendor, the unblinded, signed scrip is passed with the EPO and sent on to the issuer.

The method of moving scrip from one vendor to another presents a privacy problem by revealing the customer's past shopping behavior. It is possible to hide the details of previous vendors by passing scrips from one vendor to another via a blinding site. The customer agent will first contact the blinding site, and request the scrip to be fetched from its current site. The vendor will then retrieve the scrip from the blinding site, and will receive no information about where the scrip was prior to the blinding site.

## 3.5 Vendor Fraud Control

We have shown how customers can be prevented from double spending, even when colluding. However, floating scrips place their trust on vendors holding and transmitting scrip. A malicious vendor is able to reissue scrip to multiple vendors at the base scrip value. These funds can then be spent repeatedly at each of the vendors. In contrast to offline protocols that guard against buyer fraud, vendor fraud control is a much more realistic proposition. NetCents employs a probability based verification system to detect a malicious vendor before they are able to "profit" from their crime. A vendor only benefits when he successfully double spends funds. However, the criminal vendor is caught when the issuer receives two payment notifications with overlapping balances against the same scrip.

NetCents relaxes the policing requirements to take into effect the transaction size and the vendor's liability. Consider charging a new vendor a $200 setup fee. In this case, vendor crime will only be profitable if the vendor is able to gross more than $200 for its crime. By extending the model to take into effect a vendor's accounts receivable (from the acquirer) and its expected future income (as a product of its monthly cash flow and the number of months it has been in service) a vendor liability amount can be determined and stored in the vendor certificate. It is then only necessary to police a vendor to an extent where a vendor is better off continuing honestly rather than trying to cheat.

A scrip can travel via many vendors without central notification. As a scrip moves from one vendor to another an associated liability value, L, is set to the minimum of the scrip's liability and the current vendor's liability. The value of L thus corresponds to the liability of the least trusted vendor visited. This value is used to determine how often the issuing authority should monitor the scrip and vendor. The issuer is notified of the purchase at a probability of KP/L, where P is the purchase price and K is a constant.

K must be chosen such that the probability of a vendor benefiting from the crime is less than some desired fraction. Using probability theory as the base and selecting a probability of success at less than 50%, the following inequality is used to determine K:

$0.5 \geq (1 - K*P/L)^{L/P} + K(1 - K*P/L)^{L/P-1}$

Depending on the price, and the scrip liability, K must be chosen such that the right-hand side of the equation is always less than 0.5. To solve for K as an equality, we are left with a multivariate function that requires an iterative search to solve. Instead, we have arbitrarily chosen the value 2 as K, since this yields a fast answer, and the formula equates to below 0.5 for any P or L. The risk is the highest at low P/L (i.e. small valued transactions against well-trusted scrip), where the maximum probability of breaking even is 0.46. For larger P/L, the function is strictly decreasing.

Consider a vendor that has requested a scrip from a site with a liability of $1000 to satisfy a 5 cent transaction. The certifying bank would be contacted at a probability of 2*0.05 / 1000, or 0.0001. Since the probability of central authentication is so low for the scale of such transactions, the aggregate cost is negligible to the banks as well as the vendors. In order for a criminal vendor to benefit in the crime, it would have to succeed in making at least 20,000 five cent purchases without two separate vendors verifying the integrity of the scrip with the central authority. Using basic probability theory, a malicious vendor, in this scenario, would have a less than 40% chance of profiting from the crime (i.e. the crime does not pay).

When an issuing authority detects double spending, then the NetCents root server is presented with the evidence. The root server verifies the evidence and contacts the vendor and the vendor's acquirer. If the evidence is found to be conclusive, then a full broadcast is made to all acquirers. The acquirers, in turn, inform all of their merchants of the malicious vendor. The vendors will refuse to transfer scrips from the malicious vendor, unless it is passed via the scrip's issuing authority.

The onus is on the acquirer to manage vendor liability, policing and punishment. In the event of a malicious vendor, the acquirer is responsible for the double spent currency up to the liability amount of the vendor. If, against odds, the double spent currency exceeds the vendor liability then the claiming parties will not be fully compensated. This limit ensures that a colluding set of vendors that together have circumvented the probability central notification scheme cannot profit from colluding.

In practice, policing can be relaxed for online content that has no palpable value associated with it – such as an online article. Whereas access to an online article to a mass of colluding customers (as is possible in an offline protocol) does have an economic value associated with it, repeated downloads by one malicious vendor fails to gain from the crime. However, content with a tangible value, such as a query against costly data stores, needs to be guarded closely.

## 3.6 Currency Conversion

The importance of divisibility and the ability to handle fractions of pennies is exemplified in currency conversion. Unlike coin based payment systems that struggle to provide divisibility, NetCents' use of signed scrip balances supports payments to arbitrary precision. A NetCents scrip has a specific currency attached to it and a transparent online currency exchange mechanism is defined to handle cross-currency transactions. Since a currency conversion will typically not translate to an even number, it is important to be able to support fractions of pennies. For example, an item listed at two U.S. cents would translate to 3.52 German pfennings (@ 1.7664 marks/dollar). For a vendor to convert between currencies, the acquirer provides a periodically signed currency conversion table. The currency conversion table serves as an agreement between the acquirer and vendor where the acquirer agrees to the exchange rates for a set period of time. The acquirer would presumably set the conversion rate at a level that would protect it against unfavorable currency fluctuations.

The desired currency is included in the initial quote request message. The vendor performs the conversion on the fly and returns the quote in the customer's native currency. For the above example, a German account holder will be asked to accept a payment of 3.52 pfennings (U.S. 2 cents). The vendor, in turn, can claim the U.S. 2 cents from the acquirer by presenting the EPO spread, and referring to the currency conversion table.

# 4 NetCents Properties

In this section we discuss the properties of the NetCents protocol.

## 4.1 Atomic Properties of Transactions

The NetCents protocol can be shown to be money and goods atomic. It does not support Tygar's notion of certified delivery because of the initial clear-text transfer of item description, identifier and price from the vendor to the customer. The protocol specification can be extended to support certified delivery by requiring a signing operation by the vendor [Pou97]. However, this is a computationally expensive process that is not suitable for a low-cost payment scheme such as NetCents.

## 4.2 Security

NetCents is secure against an adversary who can intercept, destroy, modify and replay messages. Replay and double-spending attacks are ineffective because of the inclusion of the post-transaction balance and vendor identifier within the signed EPO and because of the vendor policing algorithm described earlier. Vendors cannot double charge a customer because the EPO contains a scrip balance, instead of a payment amount. A vendor cannot alter the amount charged for goods as they cannot generate a valid, signed EPO (this requires the customer's private key).

A man-in-the-middle can effectively raise the quote given by a vendor to a customer. However, if the customer agrees to pay this amount, the signed receipt will made out to the vendor and is of no use to the man-in-the-middle.

Only the legitimate owner of a scrip can cause it to be transferred within the system since the relocation process requires a signed EPO containing the balance and destination vendor. Malicious vendors, issuing authorities, etc therefore cannot cause a denial-of-service attack based on clogging the system with bogus messages. Only a customer can clog a system, and this will occur with legitimate EPOs generated by the customer. The cost of detecting a bogus message is the cost of an EPO signature verification, which is inexpensive by design.

The initial registration of a user with an issuing authority is vulnerable to attack. This process requires the transfer of sensitive information and should therefore pass through a secure, protected channel. We recommend that the registration procedure be hosted by a dedicated server, separated from the operational services of the issuing authority.

## 4.3 Cost

From an issuing authority's point of view, the system is inexpensive to implement and oversee. The issuer sells NetCents scrips to customers and distributes these scrips to vendors in response to customer requests. The issuer is updated offline when the vendors cash in their receipts. Beyond the minimal error correction and vendor policing overhead, the issuer's mission-critical workflow is independent of the transaction size. Thus, there is little incentive for an issuer to impose a minimum transaction fee. Acquirers are required only for offline batch processing and possible vendor revocation.

## 4.4 Scalability

The NetCents system (and protocol) was designed to be scalable. The SET protocol was followed in creating a trusted, interoperable network of issuing authorities and acquirers. We expect the system to be truly scalable to a global reach given its distributed nature, the low vendor verification costs, and even lower bank involvement in transactions. The only central server is the NetCents root certificate server, which signs certificates for issuing authorities and acquirers. Fortunately, this server is not part of a payment or clearing process, and thus, high availability and fast response is not a necessity. All other parties, including issuing authorities, acquirers, arbiters and blinding sites, can be arbitrarily added in response to market forces.

In a payment transaction, the only possible single point of failure comes from issuing authority downtime. Only the funds already distributed to vendors are accessible when the issuing authority is down. Unlike Millicent, however, funds can be moved between vendors to facilitate payments even when the issuing authority is inaccessible. Online arbiters can be added to the system as needed, and the system can continue to function even with complete arbiter downtime, since a complaint is not time sensitive. Similarly, blinding sites can be added as needed. If all blinding sites are unavailable, then the scrip should be passed via the bank in order to ensure customer privacy.

# 5 NetCents Implementation

## 5.1 Implementation

A prototype implementation of the NetCents protocol was built in order to test the system security and performance. The prototype runs on Intel PCs using Microsoft software technology. Off-the-shelf products were favored in order to ease development time and to make the system less proprietary.

The Client Agent evolved from a simple iterative *ping*, to test round trip latency and server throughput, to a scripted sequence of calls, and finally to a script based analysis tool that gathered detailed statistics at specified usage loads. The scripting capability allowed the development of a set of predetermined calls to NetCents vendors, issuing authorities and acquirers, and facilitated the modeling of a functioning NetCents payment environment. Transaction system practice has taught us that maximum server throughput alone is insufficient to determine acceptable loads in lieu of latency [LZGS84]. Requests are unlikely to arrive at a constant frequency, and the system must be able to gracefully handle bursts of calls. The roundtrip latency is thus calculated as the average latency over a period of time, on requests that arrive at a specific average frequency.

The NetCents issuing authority and vendor servers are built on Windows NT server technology – Windows NT Server 4.0, Internet Information Server 3.0 (IIS), and Microsoft SQL Server 6.5. The NetCents service itself is designed as an ISAPI extension. This is specific to IIS and differs from a CGI call by the fact that an ISAPI extension is a library loaded into the same address space as the server, whereas a CGI program is a fully separate process.

The issuing authority server handles requests for initializing a customer, issuing and signing scrip, and transmitting scrip to vendors when requested. For the purposes of this prototype, all NetCents requests are conducted as extended URL calls with the EPOs appended in hexadecimal notation after the URL, and replies are read as files. The vendor was given the functionality to respond to the following requests: scrip fetch process from another vendor or a bank; scrip release to another vendor; and the payment request.

## 5.2 Experimental Setup

The prototype NetCents system was set up on a 10 Mbits/s Ethernet network running Windows NT networking. One bank and two vendors were set up on three Intel Pentium Pro/180 servers on Windows NT 4.0 and IIS. The prototype system was controlled by requests from one client on a Pentium 166 also running NT 4.0. Unfortunately, the network setup was not sufficiently fast to test high load server throughput, as is evident from the low server CPU utilization. Customer requests could not be initiated on the vendor server since the client application is resource intensive and would have interfered with the server performance. The SQL server was found to consume the majority of the processing time.

## 5.3 Experimental Results

The first experiment is a simple test to measure round-trip latencies for various NetCents system calls. Its purpose is to demonstrate the base performance of the test environment. Then, the individual NetCents services are called to test the computational cost of various functions.

Table 2 shows the round trip latencies for the tests. The three first measurements – file request, CGI call and ISAPI call – demonstrate the effectiveness of the ISAPI extensions over CGI calls. However, the low server load for file request and ISAPI calls shows that throughput was not limited by the server CPU but by the test environment, possibly the network, network cards, or protocol processing at the client. Thus, Table 2 does not reveal effective throughput times.

| | Round-trip latency (ms) | Server load |
|---|---|---|
| HTTP request of 256 byte file | 9.9 | 30% |
| CGI request with 256 byte reply | 44.6 | 84% |
| ISAPI DLL call with 256 byte reply | 17.3 | 38% |
| Payment against existing scrip | 18.2 | 34% |
| Scrip fetch from a bank/merchant | 31.3 | 55% |
| Payment with scrip fetch | 63.9 | 31% |

Table 1: NetCents vendor server performance

Raw EPO signing and verification costs are very efficient when running on a stand-alone system. The RSA based signature scheme, with a 512-bit modulus and a public key of 3, yielded a signature speed of verification rate of 0.3 ms on a Pentium Pro/180 server. The most expensive component of vendor operation is the task of signing outgoing scrips, which was measured at 34 ms on the same server. Assuming a 90% hit rate (percentage of payments that do not require scrip transfer), a vendor, running on the same system, requires on average 3.7 ms per payment (not including communication and database overhead). Customer performed receipt signing is sufficiently fast at 180 ms on a Pentium/75.

However, when put into practice in an ISAPI extension, the overhead of the slow network, TCP/IP packet handling, the Internet server, and SQL database calls greatly effect the round-trip latency. Table 2 shows the observed latencies for a payment against an existing scrip (18.2 ms), a scrip fetch from an issuing authority or another vendor (31.3 ms), and a payment which requires a scrip to be fetched (63.9 ms). Again, the server load is far from its peak, and thus the maximum throughput is higher than the experiment would suggest.

Figure 3 depicts the observed average round-trip latency for purchases at varying server loads, and for different hit rates. The hit rate has a large impact on the server throughput, since a miss would result in a scrip fetch procedure involving three message exchanges: the first to inform the vendor from where to fetch the scrip; the second to actually fetch the scrip; and the last to pay against the transferred scrip. Furthermore, a fourth message is sent at a random time to fetch another scrip from the vendor, in order to keep the number of scrips in equilibrium, for the sake of our analysis.

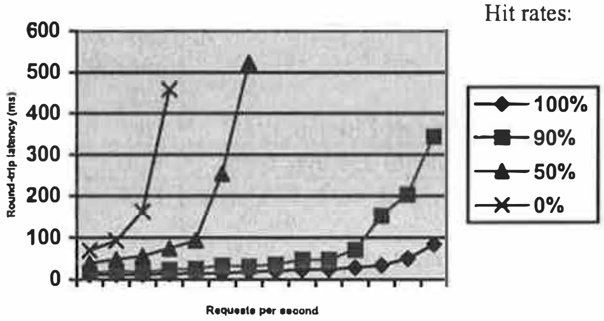


Figure 3: Average payment latency compared to the frequency of requests for various hit rates.

The graph shows the importance of modeling expected purchasing behavior in scrip relocation policies. When all payments are against scrip that reside on the vendor, then the vendor can support up to 120 payments per second (with an average latency of 371 ms) with our experimental hardware setup. For hit rates of 90% the system can comfortably handle 65 requests per second (at 200 ms). The system handles 45 and 30 payments per second for hit rates of 75% and 50% respectively. A vendor that never receives repeat buyers can only support 18 purchases per second.

## 6 Discussion

In this section we discuss some of the implementation issues with NetCents and put the NetCents protocol in the context of several existing payment protocols. Section 6.1 describes the integration of NetCents and SET. Section 6.2 compares the NetCents protocol with many popular protocols, including Millicent.

### 6.1 Integration with SET

In designing NetCents, the SET security and banking hierarchy was followed in order to allow for possible future SET interoperability. SET is an open protocol, and it can be built on top of the NetCents protocol. SET supports multiple brands, such as a VISA or Mastercard brands. NetCents can be built as another brand extension, with differences in the underlying payment mechanism. The two protocols could exist in unison, with NetCents handling all transaction of small monetary value, and SET with a credit card organization covering the higher end of the spectrum. The same public keys can be shared and the account services can be combined into one.

### 6.2 Putting NetCents in Context

In Section 2.1 we identified several desirable properties of electronic payment protocols. In this section, we evaluate NetCents against these properties. Table 2 presents the results of this comparison of NetCents with respect to the following properties and attributes.

We begin by evaluating the size of payments supported by a payment protocol, where *large payments* value exceed five dollars, *small* (to medium) *payments* range from 25 cents to five dollars, and *micropayments* range from 25 cents to fractions of a penny. The larger the range of payments that can be handled the more flexible, and real-world applicable, a protocol is.

The speed of operation of the protocol depends in large part on the speed of vendor validation. A protocol should support *fast validation* of payment deposits at a vendor. This in tum implies that public key message signing and decryption at the vendor should not be allowed due to their computational load. Given our implementation results, we postulate that a typical Pentium Pro 200 server should be able to validate at least 20 transactions per second.

One thing that limits the use of a protocol is its reliance on *customer specific hardware*. A payment system that requires users (customers) to have additional hardware will find limited deployment among potential customers.

An electronic payment system should also satisfy the ACID properties identified in section 2.1. Minimally, the protocol should be *money atomic*, in that a transaction either completes fully or does not complete at all. A protocol will be strengthened by being *goods atomic*, so that a purchase including both goods and money transfer will complete fully or not all. Finally, *certified delivery* will ensure that the protocol is goods atomic and allows proof of *what* was delivered. These attributes of a protocol will allow increased customer and vendor trust in the protocol itself.

Another attribute required for trust in the protocol is *protection from double spending*. This provides the vendor with additional assurance that they will be paid for purchases made. This also provides a level of assurance to the bank that the cost of fraud within the system will be minimized.

We also consider several characteristics of the implementation of the protocol. A good protocol will be *scalable* to a worldwide implementation, and *distributed*, to remove its dependence on a single central server. The payment means should be *divisible* to support multiple denominations and payment values. As with hard currency, we desire that electronic currency be *transferable* between users. The protocol itself should be *interoperable*, meaning that it supports multiple currencies and payment institutions.

Payment protocols may also ensure *partial anonymity*, where a buyer's identity is hidden from the vendor but not the bank, or *full anonymity*, where the buyer's identity cannot be associated with a purchase by either the vendor or the bank.

Finally, a protocol should implement *non-repudiation*, so that neither a vendor nor a customer can repudiate a transaction after the fact.

| Property | CyberCash | NetBill | DigiCash | SET | MiniPay | Agora | PayWord | Anonymous offline | Smart Card | Millicent | **NetCents** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Large-payments | ✓ | ✓ | ✓ | ✓ |  |  |  |  | ✓ |  | ✓ |
| Small-payments | ✓ | ✓ | ✓ |  | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Micro-payments |  |  |  |  | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Fast vendor valid'n. | ✓ |  | ✓ |  | ✓ |  | ✓ | ✓ | ✓ | ✓ | ✓ |
| No customer HW | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |  | ✓ | ✓ |
| Money atomic | ✓ | ✓ |  | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |  | ✓ |
| Goods atomic |  | ✓ |  |  |  |  |  |  |  |  | ✓ |
| Certified delivery |  | ✓ |  |  |  |  |  |  |  |  |  |
| Double spending protection | ✓ | ✓ | ✓ | ✓ |  |  |  |  | ✓ | ✓ | ✓ |
| Scalable |  |  |  | ✓ | ✓ |  | ✓ | ✓ | ✓ | ✓ | ✓ |
| Distributed |  |  |  |  | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Divisible | ✓ | ✓ | ✓ | ✓ |  |  |  |  | ✓ | ✓ | ✓ |
| Transferable |  |  | ✓ |  |  |  |  |  | ✓ |  | ✓ |
| Interoperable |  |  |  | ✓ | ✓ |  |  |  |  |  | ✓ |
| Partial anonymity | ✓ | ✓ | ✓ |  | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Full anonymity |  |  | ✓ |  |  |  |  | ✓ | ✓ |  | ✓ |
| Non-repudiation |  | ✓ |  |  |  | ✓ |  |  | ✓ |  | ✓ |

Table 2: Comparative analysis of payment protocol properties

## 7 Conclusions

We have introduced the NetCents micropayment protocol and demonstrated our claims that it is low-cost, scalable and secure. Previous payment protocols have traded off security for efficiency, introducing vulnerabilities to double spending, and lacking non-repudiation and anonymity. NetCents meets the requirements that we identified in Section 2.1, that is, it allows a full range of payment sizes, implements fast vendor validation, does not require additional hardware, is money and goods atomic, provides protection against double spending, implements non-repudiation, is scalable and distributed, supports multiple denominations of scrip and payment values, supports the transfer of funds between users, supports the use of multiple currencies and allows for partial and full anonymity of customer purchases.

With the NetCents protocol we have introduced the notion of floating funds that are passed from one vendor to another, following a customer's shopping tendencies. This distributed movement of funds removes the reliance on central servers in the course of a payment, making the protocol both fast and inexpensive. NetCents incorporates cryptographic functionality that makes the system secure against customer fraud; information in a transaction conclusively shows proof of payment. This proof can be used by an online arbiter for minor dispute resolution and for ensuring proper delivery of purchased goods. NetCents includes options for anonymous funds using blind signatures. Since the protocol treats money in bulk, NetCents offers a faster anonymous payment mechanism than individually signed coin based schemes.

NetCents' unique treatment of funds, in the form of floating scrips, enables distributed operation and payments without the involvement of a third party. The use of floating scrips has many benefits. First, the cost of handling a payment is limited to the computational load on part by the vendor. Since the bank is not required in the course of a transaction, then it need not impose a minimum per transaction cost to pay for the verification service. This enables vendors to sell wares at arbitrarily low costs ranging down to fractions of pennies. Second, by eliminating the bank from the transaction, the payment round-trip latency is reduced, and a central bottleneck is removed. Third, unlike offline payment protocols, NetCents is secure against customer fraud, and in particular, prevents double spending of funds.

## Bibliography

**[Br95]** S. Brands, "Proposal for an Internet cash system", Proceedings of the Internet Society; Symposium on Network and Distributed System Security, San Diego, CA, 1995.

**[CHTY96]** J. Camp, M. Harkavy, J. D. Tygar, B. Yee, "Anonymous Atomic Transactions", The Second USENIX Workshop on Electronic Commerce, Oakland, CA, 1996.

**[CST95]** J. Camp, M. Sirbu, J. D. Tygar, "Token and Notational Money in Electronic Commerce", The First USENIX Workshop on Electronic Commerce, New York, NY, 1995.

**[Ch95]** D. Chaum, "An Introduction to ecash", DigiCash, http://www.digicash.com, 1995.

**[Ch82]** D. Chaum, "Blind signatures for untraceable payments", Advances in Cryptology: Crypto '82 Proceedings. Plenum Press, 1983.

**[CFN88]** D. Chaum, A. Fiat, M. Naor, "Untraceable electronic cash", Advances in Cryptology: Crypto '88, Lecture Notes in Computer Science, no. 403, Springer-Verlag, 1988.

**[CTS95]** B. Cox, J. D. Tygar, M. Sirbu, "NetBill Security and Transaction Protocol", The First USENIX Workshop on Electronic Commerce, New York, NY, 1995.

**[DEC95]** Millicent, http://www.Millicent.com

**[GaSi96]** E. Gabber, A. Silberschatz, "Agora: A Minimal Distributed Protocol for Electronic Commerce", The Second USENIX Workshop on Electronic Commerce, Oakland, CA, 1996.

**[GR93]** J. Gray, A. Reuter, "Transaction Processing: Concepts and Techniques", Morgan Kaufmann Publishers, San Francisco, CA, 1993.

**[HalB95]** P. M. Hallman-Baker, "Micro Payment Transfer Protocol (MPTP) Version 0.1", W3C Working Draft, November 22, 1995, http://www.w3.org/pub/WWW/TR/WD-mptp.

**[HSW96]** R. Hauser, M. Steiner, M. Waidner, "Micro-Payments based on iKP", August 1996, http://www.zurich.ibm.com/iKP_references.html

**[HeYo96]** A. Herzberg, H. Yochai, "Mini-Pay: Charging per Click on the Web", http://www.ibm.net.il/ibm_il/int-lab/mpay.

**[LZGS84]** E. D. Lazowska, J. Zahorjan, G. S. Graham, K. C. Sevcik, "Quantitative System Performance; Computer System Analysis Using Queueing Network Models", Prentice-Hall, New Jersey, 1984.

**[Man95]** M. S. Manasse, "The Millicent protocol for electronic commerce," The First USENIX Workshop on Electronic Commerce, New York, NY, 1995.

**[Pou97]** T. J. Poutanen, "Metcents Protocol for Inexpensive Internet Payments", 1997. http://www.ee.ryerson.ca:8080/~hhinton/netcents

**[Sch96]** Bruce Schneier, Applied Cryptography; Second Edition", John Wiley & Sons, USA, 1996.

**[RiSh96]** R. L. Rivest, A. Shamir, "PayWord and MicroMint: Two simple micropayment schemes", 1996, http://theory.lcs.mit.edu/~rivest/RivestShamir-mpay.ps.

**[RSA79]** R. L. Rivest, A. Shamir, L.M. Adleman, "On Digital Signatures and Public Key Cryptosystems", MIT Laboratory for Computer Science, Technical Report, MIT/LCS/TR-212, Jan 1979.

**[SET97]** Mastercard, Visa, "SET 1.0 – Secure Electronic Transaction Specification", May 31, 1997, http://www.mastercard.com/set.html.

**[Ty96]** J. D. Tygar, "Atomicity in electronic commerce", Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing, pages 8-26, May 1996.

**[TW96]** J. D. Tygar, A. Whitten, "WWW Electronic Commerce and Java Trojan Horses", The Second USENIX Workshop on Electronic Commerce, Oakland, CA, 1996.

# The Auction Manager: Market Middleware for Large-Scale Electronic Commerce

Tracy Mullen and Michael P. Wellman
*Artificial Intelligence Laboratory*
*University of Michigan*
*1101 Beal Avenue*
*Ann Arbor, MI 48109-2110 USA*
{mullen,wellman}@umich.edu
http://ai.eecs.umich.edu/people/{mullen,wellman}

## Abstract

As the number and diversity of electronic commerce participants grows, the complexity of purchasing from a vast and dynamic array of goods and services needs to be hidden from the end user. Putting the complexity into the commerce system instead means providing flexible middleware for enabling commerce within different commercial communities.

In this paper, we present one such commerce middleware component — an Auction Manager designed to simplify and automate both the creation of new markets and the matching of users to existing markets. The Auction Manager determines which markets are appropriate for a given buyer or seller using market-specific inference rules applied to the current market offerings. We also show how these same inference rules can be used by the Auction Manager to automatically compose and decompose market offerings to respond to changing conditions within the marketplace. Finally, we describe how the Auction Manager provides a focal point for expressing policy decisions such as how much to charge for starting and running auctions, as well as who and when to charge.

## 1 Introduction

At the level of individual computers, operating systems shield applications from system details by providing direct access to general system services. In the same way, middleware supports transparency in remote services provided over networked information systems. In the realm of electronic commerce, the role of middleware is to support the basic operations that might take place within a commercial interaction. Although the varieties of commerce activities are legion, as long as they share some fundamental components, reusable middleware services can offer substantial leverage. For example, the fundamental step of executing an exchange transaction is common to many contexts, and so general infrastructure supporting this operation will have many uses.

A significant burden of middleware, however, is to support a diversity of commercial communities—equity and commodity markets, manufacturing supplier chains, mail-order retail, and publishing, just to name a few. Each such community comes with its own particular vocabularies, institutions, and conventions, evolved for its own purposes and entrenched to varying degrees. This suggests that despite any fundamental commonalities, commerce middleware services need to be flexible and extensible to support (at least) the range of practices we find in the natural commercial world.

An agent wishing to participate in a commercial interaction (an *exchange* of some good or service, which we generically call a *good*) must face each of the following questions on entry to a commerce environment:

1. How can I describe what I want to exchange?
2. Where can I exchange the good and under what terms?
3. Who can I exchange with?
4. How can we execute the transaction?

Addressing each of these questions constitutes a step in a comprehensive commerce process. More specialized commerce environments, such as business-to-consumer commerce, might include additional steps such as promotion. The role of commerce middleware is to serve agents taking these steps. One can conceive of reusable support services targeted at individual steps (or parts of steps), or those spanning multiple steps [9].

Probably the most well-defined of these steps is the fourth—executing the transaction—and indeed, the largest share of middleware services termed "electronic commerce" primarily address this step. Early payment protocols, such as those developed by First Virtual (http://www.firstvirtual.com/) and Digicash (http://www.digicash.com/), as well as proposed generic frameworks [11], clearly fall in this category. As pointed out by MacKie-Mason and White [14], even this relatively circumscribed step presents interesting design choices along many dimensions.

Probably the least well-defined step is the first—describing the good. Ongoing work in developing descriptive metadata for e-commerce ranges from formalizing license terms descriptions [19] to more ad hoc discussions of XML-based content definitions such as CommerceNet's XML exchange (http://www.xmlx.com).

In this paper, we focus on middleware services for step two. However, we build on the growing presence and increasing understanding of the value of on-line auctions [20] for step three—setting the terms of an agreement and matching buyers with sellers.

We describe here an *Auction Manager*, a commerce middleware component designed to simplify and automate both the creation of new markets and the matching of agents to existing markets. Some standalone products aim to provide some of this function. For example, shopping agents, such as Bargain Finder (http://bf.cstar.ac.com/bf) and Jango [6] (http://www.jango.com), provide consumers with compilations of Internet vendors' prices for products such as music CDs or software. In contrast, middleware for this type of task is more infrastructural, serving individual functions as generally as possible, in the context of supporting an overall commerce process. In particular, generic infrastructure for step two could not assume a market model where vendors announce a fixed price for consumers to take or leave. Rather, there might be many modes of negotiation, which market-matching services should take into account in identifying potential matches.

The Auction Manager operates within a dynamic environment, by matching descriptions of goods to existing markets and, when appropriate, creating new markets. While *market* is often used as a non-technical term, we use it here to refer specifically to an auction and its authorized types of participating agents (e.g., stock markets permit only certain authorized broker agent types to participate).

Implicit in the Auction Manager's support for market-matching operations are questions of market policy. For example, when and for what goods should new markets be started? How does the system account for market creation costs? How are community rules, norms, and objectives (if any) expressed—through regulations or incentives?

The Auction Manager was built as part of the University of Michigan Digital Library [2] (UMDL) commerce infrastructure. Whereas UMDL's goal is the provision of library services to library users, the explict realization of that goal is to provide a commerce infrastructure that supports the process of describing, locating, and negotiating for a wide variety of information services.

However, this commerce infrastructure is not restricted or specific to digital libraries. Since UMDL cannot know at design-time what services will be available in the future or what the best negotiation mechanisms are for any given situation, its languages and protocols have been designed for flexibility and extensibility. One of the advantages of this flexibility is that UMDL provides a testbed, allowing us to experiment with different mechanisms in different contexts.

Within the UMDL, library exchanges focus on a particular kind of good—information services. Therefore, in the next sections we often refer to goods when describing abstract exchanges, while referring to particular information services in our more concrete examples.

In the next section, we describe our generic commerce infrastructure. Our main aim is not to describe the existing UMDL architecture[1] in detail, but to give an overview of the context in which the

[1] Ongoing UMDL work can be found at http://www.si.umich.edu/UMDL.

Auction Manager operates. In Sections 3 and 4, we focus on the commerce middleware services of the Auction Manager.

## 2 Commerce Infrastructure Overview

One part of the commerce infrastructure is the interaction framework: description languages for goods, and negotiation and exchange protocols, which we describe in Sections 2.1 and 2.2. The second part comprises the various infrastructure components, described in Section 2.3, which simplify and automate the use of these languages and protocols.

### 2.1 Description languages for goods

Description languages, or ontologies [8], permit a large number of potential goods and their attributes to be captured as a taxonomy. Using these structured objects, we can identify and reason about classes of goods [4]—a powerful base for reusable automation.

For example, one common library service is that of answering queries. A query planning service will, based upon the user's query, search the appropriate collections for relevant items, possibly employing a thesaurus or other indexing service. Query planning services may differentiate themselves through the different attributes associated with them, such as target audience (Professional, High School, Middle School), topic (Science, Art, Literature), and recommending organization (ACM, IEEE, National Education Association). Figure 1 shows a simple taxonomy of possible query planning audience and topic attributes.

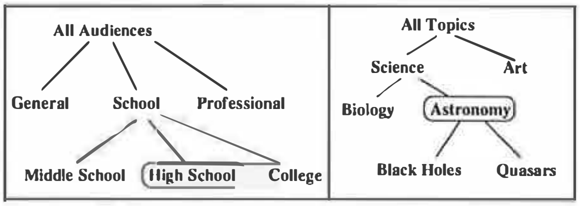


Figure 1: Query Planning: audience and topic attributes

Clearly *High School* audiences are just a particular kind of *School* audiences and *Astronomy* topics are a subclass of *Science* topics. Thus, a service that can respond to queries about High School Science will also be able to respond to queries about High School Astronomy. We discuss the opportunities, and special considerations, involved in using these sort of inference rules to match agents with markets in Section 3.1.

### 2.2 Negotiation

Given the enormous number of potential goods and the dynamic conditions of the marketplace, there is a concomitant need for a variety of negotiation mechanisms. For example, if a seller has a monopoly good in the current marketplace, the same good would be sold under very different terms than in a more competitive marketplace.

Auctions, which are simply a set of rules for determining a price and/or allocation based on a bidding protocol [16], provide a very flexible negotiation framework—each different auction institution can have a large effect on trading outcome. In effect, auctions are just another service—they provide matching and price setting for buyers and sellers. Auctions also promote automated negotiation through the following characteristics:

- **Mediated**
  Every buyer does not have separately find and contact every seller.
- **Price, not barter**
  Price minimizes and simplifies communication.
- **Formal**
  Standardized, structured offers and auction rules simplify communication as well as processing of offers.

We capture information about the different auction rules and protocols in a compact, reusable manner by using parameterized auction descriptions [18, 25]. For example, an auction's attributes include how often it clears, its price determination rules (e.g., first price, second price), the allowed number of buyers and sellers, as well as what information is publicly available. A Vickrey auction can be described as having a price determination rule of second price, many buyers and one seller, and where the publicly available information does not include the bids (since it is sealed-bid). By changing these kind of auction parameters, we would expect to get a wide variety of behaviors and outcomes. These parameterized auction descriptions provide a description language, or standard vocabulary, for auctions.

### 2.3 Infrastructure components

Infrastructure components simplify, augment, and automate the use of the underlying goods and services description languages. They do this by encapsulating reusable services common to different steps in the commerce process. We describe our commerce infrastructure components below, as they would be invoked in a typical scenario, diagrammed in Figure 2.

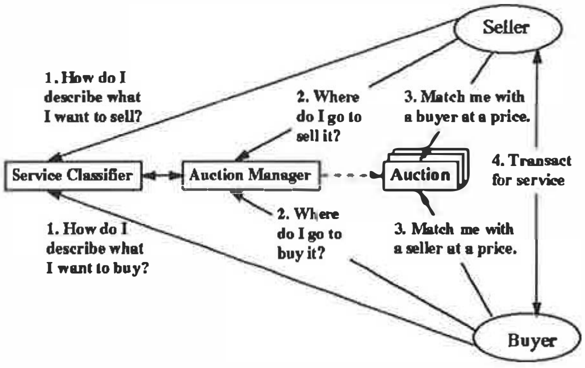

Seller
1. How do I describe what I want to sell?
2. Where do I go to sell it?
3. Match me with a buyer at a price.
Service Classifier
Auction Manager
Auction
4. Transact for service
1. How do I describe what I want to buy?
2. Where do I go to buy it?
3. Match me with a seller at a price.
Buyer

Figure 2: Infrastructure components

Initially, every agent or auction registers with the Registrar (not pictured) and receives an agent-id or auction-id, which uniquely identifies it and is used for communication within the system.

Step one in the information exchange process is for a buyer or seller agent to describe what good it wishes to exchange. For example, agents might describe the particular kind of query planning service they are providing or seeking in terms defined in some standard vocabulary. An agent sends this service description to the Service Classifier [22]. The Service Classifier classifies the service within a comprehensive taxonomy of information services and responds with a service label. The service label acts as a tag identifying a classified service within the system.

In step two, agents want to locate auctions where they can buy or sell the service. An agent sends the service label to the Auction Manager, adding information about particular auction attributes if they wish. The Auction Manager informs an agent of existing auctions for that service or, if necessary, creates a new auction.

In step three, each agent chooses one or more auctions to participate in. Buyer and seller agents are matched and a transaction price is set according to the rules of the auction. In the last step, the two agents transact and exchange the service.

## 3 Market Management Services

In this section, we focus on the market management services encapsulated by the Auction Manager. In the process of building and using the Auction Manager service within UMDL, we identified three core services: market matching for buyers and sellers, notification of new markets, and data collection and information dissemination. We describe each of these services below. For our examples, we use the query planning services introduced in Section 2.1.

### 3.1 Market matching

In a large-scale and dynamic commerce environment, the Auction Manager's role of matching agents with the appropriate markets is nontrivial. If the exact market that the agent has requested exists, then all the Auction Manager has to do is return that market. However, given the wide variety of possible service and market descriptions, an exact match may not exist. Even if an exact match exists, the agent may wish to know about all markets in which it could trade—there may be tradeoffs between price and quality that only the agent can make. For example, suppose there are two query planning markets: one in High School Science, and one in High School Astronomy. If the High School Astronomy market is smaller and the quality higher (because it is more specialized), then the price may be higher. An agent wanting High School Astronomy query planning has to make a choice between the higher quality of the Astronomy market versus the lower price of the more general Science market.

Given that we can express these different goods and markets in some description language, how do we reason about them in a market context? In particular, what kinds of market-specific operators and inference rules are needed?

### 3.1.1 Market-specific operators

In the query planning service example above, we have made some assumptions about what it means in the taxonomy for the topic Science to be more general than Astronomy in a market context. However, there are a number of different ways in which we could interpret this taxonomy.

Consider the difference between a Science query planning service that is composed of *bundled* Astronomy and biology query planners and one that makes *undifferentiated* queries across any and all Science sources. We can describe the *bundled* (⊗) technology as one that can be easily decomposed, or unbundled, into several subparts whereas an *undifferentiated* (⊙) technology cannot. Next, assuming that Science can be unbundled into separate Astronomy and Biology query planners, is it the buyer or seller who chooses which service will be used? We describe *buyer's choice* using the operator Ⓑ and *seller's choice* with Ⓢ.

These four logical operators on market descriptions are shown below:

| *Operator* | *Example* | *Meaning* |
|---|---|---|
| Ⓑ[class *c*] | Ⓑ[*Science*] | Buyer's choice of subclasses of Science |
| Ⓢ[class *c*] | Ⓢ[*Science*] | Seller's choice of subclasses of Science |
| ⊗[class *c*] | ⊗[*Science*] | Bundled subclasses of Science |
| ⊙[class *c*] | ⊙[*Science*] | Undifferentiated Science |

In the case of query planning services, it seems unlikely that a buyer is going to want to let the seller choose the topic (i.e., seller's choice). Indeed, within the UMDL, we make the default assumption that all query planning services are buyer's choice, considerably simplifying the search for related markets.

However, for other goods seller's choice is a reasonable operator. Consider a newspaper subscription—subscribers don't choose the articles that appear in each day's paper. In a buyer's choice article market, buyers would pick and choose among available articles, essentially creating their own customized paper. Each of these two choice operators composes individual articles into a different, more abstract, bundled article market.

The idea of bundled goods is far from new, and has been generally addressed in economic literature as an alternative technique for price discrimination [17]. By reducing the dispersion in consumer tastes across different bundles a seller can often extract more value from transactions than uniform pricing would. Although historically goods have often been unprofitable to bundle, in the case of information goods and services (with marginal costs close to zero) selling bundled goods can yield higher profits and greater efficiency than selling them separately [3]. One ongoing field trial in this area includes the PEAK project, offering Elsevier journal articles with different experimental bundles and prices [13].

What we introduce here is the ability of the buyer or seller to choose a sub-bundle of the bundled goods, as defined in Table 2. These are not the only possible ways of choosing sub-bundles; the set of market operators could be extended. For example, a ⊗[Science] is a bundle of Science subtopics, which are themselves bundles of their subtopics. We can always describe non-recursive bundles explicitly (e.g., ⊗[⊙[*Astronomy*], ⊙[*Biology*]]) or define another operator explicitly to handle such bundles.

| *Operator* | *Definition* |
|---|---|
| **Bundle** | ⊗[*superclass*] ≡<br>⊗[⊗[$class_1$], ..., ⊗[$class_n$]] |
| *Example:* | ⊗[*Science*] ≡<br>⊗[⊗[*Astronomy*], ⊗[*Biology*]] |
| **Buyer's Choice** | Ⓑ[*superclass*] ≡<br>Buyer chooses one of<br>{ Ⓑ[$class_1$], ..., Ⓑ[$class_n$] } |
| *Example:* | Ⓑ[*Science*] ≡<br>Buyer chooses one of<br>{ Ⓑ[*Astronomy*], Ⓑ[*Biology*] } |
| **Seller's Choice** | Ⓢ[*superclass*] ≡<br>Seller chooses one of<br>{ Ⓢ[$class_1$], ..., Ⓢ[$class_n$] } |
| *Example:* | Ⓢ[*Science*] ≡<br>Seller chooses one of<br>{ Ⓢ[*Astronomy*], Ⓢ[*Biology*] } |

Table 2: Market operator definitions

In the next section, we describe how to use this notation for automatic matching of buyers and sellers to markets.

| *Service Market #* | *Trade Context* | *Query Planning Service Attributes* | | |
|---|---|---|---|---|
| | | *Audience* | *Topic* | *Recommending-Org* |
| 1 | buy, sell | High School | Ⓑ[Astronomy] | unspecified |
| 2 | buy | High School | Ⓑ[Astronomy] | Astronomy Today |
| 3 | buy | Ⓑ[School] | Ⓑ[Astronomy] | unspecified |
| 4 | sell | High School | Ⓢ[Science] | unspecified |
| 5 | sell | Ⓢ[School] | Black Holes | unspecified |
| 6 | sell | unspecified | Ⓑ[Astronomy] | unspecified |

Table 1: Existing Markets 1-6

### 3.1.2 Market-specific inference rules

Finding all the potential markets for an agent to trade in requires that the Auction Manager employ not only the taxonomy of good descriptions and market operators; it also needs to know whether the agent wants to buy or to sell a particular good.

To see why this is necessary, consider two agents, one who wishes to buy a query planning service for High School audience for Astronomy topics and one who wishes to sell it. Note that in these descriptions we refer back to the simplified version of the audience and topic attributes taxonomy shown in Figure 1. In table form this service description looks like:

| *Service* | *Service Attributes* | |
|---|---|---|
| | *Audience* | *Topic* |
| Query Planning | High School | Ⓑ[Astronomy] |

We represent the Astronomy topics as buyer's choice because that makes the most sense for a query planning service. Since High School audience has no subclasses under it, it is not necessary to use an operator to describe how the subclasses are bundled. Next, suppose that Markets 1-6 in table 1 already exist. Notice that these markets may include additional attributes, such as *Recommending-Organization*, which were not specified in the agent's service description. Similarly, the markets may also lack attributes which are specified by an agent.

Both buyers and sellers could participate in Market 1, since it exactly matches the service description. However, in Markets 2-6, whether an agent can participate depends on the trading context (i.e., whether the agent wants to buy or sell the good).

In Market 2, the seller cannot participate since it has not been recommended by Astronomy Today. On the other hand, whether a service has been recommended or not is irrelevant to the buyer, so it can participate.

In Market 3, a buyer can always choose High School audience among the different kinds of School audiences offered. However, the seller's service is limited to High School audiences and it cannot participate in a School audiences market because buyers might want to select Middle School audience or Professional audience.

In Market 4, a seller can participate since Astronomy is a Science and the seller has the right to choose which kind Science it provides—in this case the seller would always choose Astronomy. However, if a buyer were to participate, the query planning service the buyer got matched with might very well be for Biology rather than Astronomy.

In Market 5, a seller who can offer any Astronomy subtopics, can simply unbundle the Astronomy subtopics and offer Black Holes separately. On the other hand, a buyer interested in Astronomy topics may not want to confine its queries to Black Holes but may be interested in Quasars also.

Lastly, in Market 6, the seller can participate in a market with unspecified audiences, since buyers haven't specified (don't care) what kind of audience the query planning service is aimed at. However, a buyer who specifically wants a High School audience cannot.

The informal rules expressed in the example above can be captured and used by the Auction Manager to automatically match potential markets.

| *Rule* | *Example* |
|---|---|
| **Generalize Attribute Class** | *Operator*$_1$*[class]*<br>→ *Operator*$_2$*[superclass]* |
| *When buying:* | Ⓑ[Astronomy] → Ⓑ[Science]<br>Ⓢ[Astronomy] → Ⓑ[Science]<br>⊗[Astronomy] → Ⓑ[Science]<br>⊙[Astronomy] → Ⓑ[Science] |
| *When selling:* | Ⓑ[Astronomy] → Ⓢ[Science]<br>Ⓢ[Astronomy] → Ⓢ[Science]<br>⊗[Astronomy] → Ⓢ[Science]<br>⊙[Astronomy] → Ⓢ[Science] |
| **Choice Operators** | *Operator*$_1$*[class]*<br>→ *Operator*$_2$*[class]* |
| *When buying:* | Ⓢ[Science] → Ⓑ[Science] |
| *When selling:* | Ⓑ[Science] → Ⓢ[Science]<br>⊗[Science] → Ⓑ[Science] |
| **Join** | ⊗*[class]* → ⊙*[class]* |
| *When selling:* | ⊗[Science] → ⊙[Science] |
| **Decompose** | ⊗*[class]* → one of ⊗[subclass] |
| *When selling:* | ⊗[Science] → ⊗[Astronomy] |
| **Unspecified Attributes** | |
| *When buying:* | unspecified → anything |
| *When selling:* | anything → unspecified |

Table 3: Some market inference rule examples

For example, the Generalize Attribute Class rule in Table 3 says that a buyer who wants topics on Astronomy, can be satisfied in a market for buyer's choice Science. Similarly, a seller who sells seller's choice topics in Astronomy, can sell in a seller's choice market for Science topics—for any customer, the seller can always choose Astronomy as the topic. By writing potential trades between markets as transformation rules, we can use forward and backward chaining to ask questions, such as what are all the potential ways that this product could be sold.

These rules model how agents can bundle and unbundle goods and choose from among bundled goods. However, there is no guarantee that these rules will terminate. In practice, both the number of markets and the existing UMDL ontology are sufficiently small that this has not been an issue. In the future, we will need to place restrictions on the rules of inference to be able to guarantee termination. Similar problems for automatic checking of security protocols using formal logics were addressed by an automatic theory-checker generator [12].

### 3.1.3 Automatic market arbitrage

Our discussion of the inference rules in table 3 above has implicitly made the assumption that selecting the appropriate good from several others is a simple operation. For example, the Generalize Operator rule implies that an agent who wants to buy a seller's choice Science query, Ⓢ[Science], can buy a buyer's choice Science query, Ⓑ[Science], and pick an arbitrary Science topic. There is clearly a potential trade between the buyer and seller, which can occur just by making an arbitrary selection between topics. If the choice is process is well-defined, it could be automated either within an individual agent or by having a specialized Ⓑ-to-Ⓢ *arbitrage agent* created on demand to link the two markets.

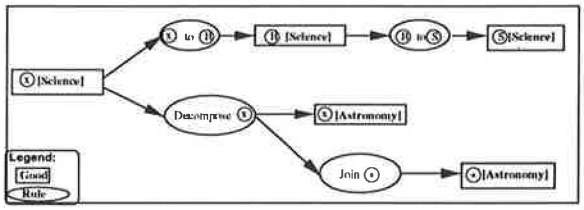


Figure 3: Applying market inference rules for seller

Figure 3 shows how these rules, using appropriate control of the rule execution process, can generate a large number of different products, based on a few simple transformations. This suggests that one-time requests for products requiring simple transformations may be more efficiently handled through automatically generated arbitrage agents or by encoding simple transformations within agents, compared to starting up an entirely new market.

## 3.2 Notification

Once a new market has been created, initially only the Auction Manager and creating agent know about it. For other agents to be aware of the new auction, either they will have to periodically check for new auctions or rely on a notification service to alert them, depending on their individual needs. The Auction Manager serves as a focal point for our notification services since it both creates markets and accepts the original service requests from agents.

Whenever an agent requests a market, it can ask the Auction Manager to notify it if any new matching markets are created. In order to determine if an agent would be interested in a newly created auction, the Auction Manager compares its service label to the service label of the newly created auction using the market matching described in Section 3.1. If the agent's service can be exchanged there, then the Auction Manager sends a notify message to the agent telling it the new auction-id.

### 3.3 Data collection and information dissemination

Markets are not only means of trading, but also a source of information about the price and other terms under which trades may be made. Of course, revealing this information may itself distort agent behavior, and whether it will turn out to be a good or bad idea to disseminate any market information remains an open question. Two sources of market data are described below.

One source is the data logged by the auctions. This includes consumer and producer bid values, time of bid arrivals, number and time of transactions, as well as clearing prices. From this information, summary data such as bidding frequency, average price, and price variance can be produced.

Another source of market data is the original service request from an agent. Sometimes the Auction Manager can exactly match an agent to a market, otherwise the agent has to participate in a near-match market. However, in the second case, the Auction Manager knows both the original service demanded and that an exact match for that service could not be found. This information may be valuable to sellers, upon being made aware of unmet demands by buyers for services, they could estimate whether it would be worthwhile be willing to invest in developing or specializing their agents to meet that demand. By collecting and providing this kind of data about current market activity and demand for services, the Auction Manager can provide agents with information on which to base their decisions about how to trade their services or when it may be worthwhile to develop new ones.

Such market data can be used offline to evaluate the effect that different market creation and selection policies had on the market configuration and resulting system efficiency and welfare. However, exactly under what circumstances this information should be made available, and to whom, depends on the commerce community and is a matter of market policy.

## 4 Market Policies

Market policies can serve to support and uphold established business practices for a community as well as account for system externalities such as market creation costs. Policies can be implemented through either rules or incentives. For example, policy issues such as information access management [1], which determine who may access what collections and under what terms and conditions, are more naturally handled as rules. On the other hand, objectives such as increasing some social welfare criteria or providing a base level of library services to all patrons, may be more naturally implemented as incentives, perhaps subsidies or taxes.

Our focus is on policies related to market management costs. Since the number of possible markets is virtually unbounded, while the amount of network resources and agent attention is not, we consider what kinds of policies need to be employed to restrict market creation and select reasonable markets. In Section 4.1, we discuss the sources of market creation costs. In Section 4.2, we discuss the problem of who should decide what new markets can be created.

### 4.1 Market creation costs

In an ideal system, where running an auction is a cost-free proposition and agent decision costs are not considered, there would be no reason not to allow any and all markets requested. However, this ideal environment does not exist; running an auction is not cost-free.

*Infrastructure costs* include the computational and network costs of actually running the auction and—even more importantly—the resultant increase in lookup and indexing costs. Clearly these costs must accounted for through a policy which can set the appropriate fees. One simple policy is to have a uniform flat fees charged for infrastructure usage. For example, an auction might either charge a single fee to the auction owner/initiator to run it for a given time period, or charge smaller per-transaction fees to individual auction participants. On the other hand, a library policy might use system-level criteria to promote auctions which the library considers beneficial to the system as a whole, perhaps by charging less for them, or running them for free. A side benefit to having an explicitly stated policy, whether system-oriented or not, is that it could be used to provide guidance when choosing among a number of potential auction specifications for a given product.

*Agent decision complexity costs* result with the increase in market choices. If an agent can buy a service from several related markets, the agent might handle it in one of two basic ways. The first is to trade simultaneously in multiple markets, facing the risk of ending up with multiple outstanding commitments. The second is to trade in only one market at a time, facing problems of market liquidity where there may be two parties willing to trade, but unable to do so since they are at different markets. Of course, an agent may make sequential offers at different markets, but this results in trading delays and will not necessarily bring all parties together, due to timing difficulties.

The Auction Manager can provide a vehicle to experiment with alternative market-creation policies by evaluating the resultant market configurations.

### 4.2 Market selection issues

Given the potentially unbounded number of markets that can be created for a partially specified service, how do we decide which market should be created? In the past, market selection has been addressed in work on incomplete markets [10, 15], product differentiation [5], and industrial organization with transaction costs [23]. This work generally involved analyzing the effects for a known and static number of markets.

When neither the exact number nor kind markets can be known ahead of time, methods for comparing and evaluating the benefits of different market configurations [7] will need to be developed. Using such comparisons, we can hope to identify general rules or incentives that promote commerce environments having increased efficiency, profits, surplus or any number of other criteria. We are currently performing small experiments to compare the efficiency and surplus for different market configurations given different buyer and seller value distributions.

In the next sections, we consider the mechanics of two particular market selection processes. The mechanics of the market selection process will have an impact on what kinds of markets can be started as well as the kind of rules and incentives which can be used. In particular, who makes the decisions about what auctions to create, is it the agents (where agents can be humans or software) or the Auction Manager? We argue that in some circumstances it may be more reasonable to have agents decide while in others the Auction Manager, and both should be supported. We also discuss the kinds of information and incentive requirements needed for each situation.

#### 4.2.1 Agents decide

Under certain circumstances, it will make more sense, or at least be more realistic, for agents to determine which markets are created. For example, when an agent represents an outside vendor, especially a brand-name vendor, the vendor may very well want to establish its own market. Also, as buyer agents and/or recommending organizations learn which information providers and kinds of services they prefer, they should be free to establish markets which capture these preferences [21].

However, allowing agents to decide which markets to create means that the externalities involved in creating an auction, such as the costs to the system and other users, must be internalized in the form of auction fees. The fees need to be set so as to provide the right incentives for agents not to abuse the auction creation mechanism. We are in the process of exploring how to set these fees. Some requirements are that the fee structure needs to be kept simple, as complicated schemes may create additional costs that overwhelm the original system costs involved. Reasonable choices include setting a small flat fee per auction transaction, a small percentage fee per transaction or a rental fee per time period that the agent who creates the auction pays.

In deciding whether to establish a new market, an agent may find it useful to consider information about related product auction prices, transaction volumes and frequency, and unmet user demands. This was discussed in Section 3.3

#### 4.2.2 Auction manager recommends

The Auction Manager can provide an auction recommendation service by using auction- and good-specific knowledge to pick reasonable auctions. Thus having the Auction Manager recommend auctions is not necessarily incompatible with having agents deciding which auctions they want to create. Agents can specify which service or auction attributes they care about and the Auction manager can fill in the rest.

Some of these defaults are independent of the particular context. For example, no one would want an auction for immediate query planning service to have a clearing time of once a day. Also, if a market is inactive for a certain amount of time it makes sense to have it deactivate itself, since it can always be restarted if necessary. However, choosing the appropriate auction(s) will generally be affected by user preferences, technology factors, and the market environment. If an agent is a monopolist then its definition of the best kind of auction will differ from one for a library agent who charges its marginal cost.

A key question is how much does the Auction Manager know about participating agents and what are the criteria that it uses to determine the best auctions. In the case of the outside vendors, we may be able to support their choice of auctions in two ways. The first is simply by using defaults to fill in partially specified auctions with reasonable values. The second is by using economic theory to set up auctions depending on what kind of agent requests it. For example, the Auction Manager can choose an auction which is incentive compatible for buyers or for sellers, but not both [24].

In the case of library agents, where the focus is more oriented towards designing simpler agents to allocate library services efficiently, the Auction Manager could have a library policy to determine which are the best auctions to create. Due to the complexity of the auction design space, a realistic library policy would have to be expressed in simple, qualitative terms such as more market efficiency is better, lower transaction costs are better, less price volatility is better, and so forth.

## 5 Conclusion

The Auction Manager is a market middleware component providing services to support automated negotiation in a large-scale electronic commerce systems. Specifically it supports this by generating and tracking auctions, matching agents to potential markets, and providing a means to notify agents when markets of interest to them are created. We have described how the Auction Manager can use market-specific knowledge to recommend auctions and how it can serve as a focal point for information collection and dissemination.

Future work involves using the Auction Manager to experiment with different auction creation policies and auction pricing policies and, based on the resultant market configurations and market data, evaluate their effectiveness in promoting specified system policies.

**Acknowledgments.** This work was supported by the NSF/ARPA/NASA Digital Library Initiative. Participants in the UMDL project have contributed to the concepts and work described here, especially William Birmingham, Edmund Durfee, Jeffrey MacKie-Mason, Anisoara Nica, Sunju Park, José Vidal, William Walsh, and Peter Weinstein.

## References

[1] William Yeo Arms. Implementing policies for access management. *D-Lib Magazine*, February 1998.

[2] Daniel E. Atkins, William P. Birmingham, Edmund H. Durfee, Eric J. Glover, Tracy Mullen, Elke A. Rundensteiner, Elliot Soloway, José M. Vidal, Raven Wallace, and Michael P. Wellman. Toward inquiry-based education through interacting software agents. *IEEE Computer*, 29(5):69–76, May 1996.

[3] Yannis Bakos and Erik Brynjolfsson. Bundling information goods: Pricing, profits and efficiency. In *Conference on Economics of Digital Information and Intellectual Property*, Harvard University, January 1997.

[4] Alexander Borgida. Description logics in data management. *IEEE Transactions on Knowl-*

*edge and Data Engineering*, 7(5):671–682, October 1995.

[5] Avinash K. Dixit and Joseph E. Stiglitz. Monopolistic competition and optimum product diversity. *The American Economic Review*, 67(3):297–308, June 1977.

[6] Robert Doorenbos, Oren Etzioni, and Daniel S. Weld. A scalable comparison-shopping agent for the world-wide web. In *First International Conference on Autonomous Agents*, pages 39–48, 1997.

[7] Robert P. Gilles, Dimitrios Diamantaras, and Pieter H. M. Ruys. Efficiency, valuation and cores in economies with costly trade. Technical Report E96-01, Department of Economics, Virginia Tech, 1996.

[8] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5/6), 1995.

[9] Robert H. Guttman, Alexandros G. Moukas, and Pattie Maes. Agent-mediated electronic commerce: A survey. In *Knowledge Engineering Review*, volume 13, June 1998.

[10] Frank Hahn, editor. *The Economics of Missing Markets.* Clarendon Press, Oxford, 1989.

[11] Steven P. Ketchpel, Hector Garcia-Molina, Andreas Paepcke, Scott Hassan, and Steve Cousins. U-PAI: A universal payment application interface. In *Second USENIX Workshop on Electronic Commerce*, pages 105–121, November 1996.

[12] Darrell Kindred and Jeannette M. Wing. Fast, automatic checking of security protocols. In *Second USENIX Workshop on Electronic Commerce*, pages 41–52, November 1996.

[13] Jeffrey K. MacKie-Mason and Juan F. Riveros. Economics and electronic access to scholarly information. In B. Kahin, editor, *The Economics of Digital Information.* 1998. (forthcoming). http://www-personal.umich.edu/ jmm/papers.html.

[14] Jeffrey K. MacKie-Mason and Kimberly White. Evaluating and selecting digital payment mechanisms. In G. Rosston and D. Waterman, editors, *Interconnection and the Internet*, pages 113–134. Lawrence Erlbaum, 1997.

[15] Michael Magill and Martine Quinzii. *Theory of Incomplete Markets, Volume 1.* The MIT Press, 1996.

[16] R. Preston McAfee and John McMillan. Auctions and bidding. *Journal of Economic Literature*, 25:699–738, 1987.

[17] R. Preston McAfee, John McMillan, and Michael D. Whinston. Multiproduct monopoly, commodity bundling, and correlation of values. *Quarterly Journal of Economics*, 104(2):371–384, May 1989.

[18] Tracy Mullen and Michael P. Wellman. Market-based negotiation for digital library services. In *Second USENIX Workshop on Electronic Commerce*, pages 259–269, November 1996.

[19] Godfrey Rust. Metadata: The right approach. *D-Lib Magazine*, July/August 1998. http://www.dlib.org/.

[20] Efraim Turban. Auctions and bidding on the internet: An assessment. *International Journal of Electronic Markets*, 7(4), 1997. http://www.electronicmarkets.org.

[21] José M. Vidal and Edmund H. Durfee. Learning nested agent models in an information economy. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3):291–308, 1998.

[22] Peter Weinstein and William P. Birmingham. Runtime classification of agent services. In *Proceedings of the AAAI-97 Spring Symposium on Ontological Engineering*, Stanford, Palo Alto, CA, March 1997.

[23] Oliver E. Williamson. Markets and hierarchies: Some elementary considerations. *The American Economic Review*, 63(2):316–325, May 1973.

[24] Peter R. Wurman, William E. Walsh, and Michael P. Wellman. Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems*, to appear.

[25] Peter R. Wurman, Michael P. Wellman, and William E. Walsh. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In *Second International Conference on Autonomous Agents*, pages 301–308, May 1998.

# Internet Auctions

Manoj Kumar (mkumar@watson.im.com)
Stuart I. Feldman (sif@watson.ibm.com)
*IBM Research Division*
*T.J. Watson Research Center*
*Yorktown Heights, NY 10598*

## Abstract

*We describe an application for auctioning goods on the Internet. A variety of commonly used auction mechanisms that are supported by the application, security requirements, and pre-auction and post-auction interactions needed to complete auction based trading are discussed. Then we present a software architecture and describe the various processes that comprise the auction application. Finally, we discuss how the delay, security, and easy collaboration aspects of the Internet will cause auctions on the Internet to be different than the traditional auctions.*

## 1. Introduction

Most business activity on the Internet is limited to publicizing the business opportunity and catalog based sales, but it will rapidly expand to include the negotiations conducted to settle the price of the goods or commodities being traded. These negotiations are currently conducted by human intermediaries through various forms of auctions, bidding systems for awarding contracts, and brokerages. The role of the intermediaries can now be performed by Internet trading applications at a fraction of the cost. Trading on the Internet allows a business to reach a larger number of potential customers and suppliers in a shorter time and a lower cost than possible by other modes of communication, and to settle business transactions with lower cost overhead in a shorter time. Hence the rapid emergence of Internet based trading applications. Lee discusses the factors behind the success of Internet auction of second hand automobiles in Japan [1] supporting our belief.

Auctioned or brokered sales are the norm in the business world for negotiating trades of large value. But consumer sales and small scale purchases typically stay with fixed prices, perhaps because of the high overhead cost of using the auction or brokerage method. The new economics of the Internet will make auctions popular in consumer and small business transactions also. Lee and Clark present economic forces underlying this transition [2]. Several success stories about Internet auctions are cited by Turban [3].

Many types of auctions are practiced in different real world situations to achieve different business objectives such as best price, guaranteed sale, minimum collusion possibility, etc. Ralph Cassady presents an extensive survey of auction practices around the world [4]. Game theoretic treatments of the different kinds of auctions can be found in [5,6,7], while some experimental results are reported in [8]. In this paper we describe the design of an Internet auction system that can support most of the auction types and other business negotiation models.

In this paper we first briefly review the requirements of an Internet auction application. Some of the important requirements are support for a wide variety of commonly practiced auctions and ease of integrating auctions with business's existing back end applications to create a completely automated trading process. Security mechanisms, based on cryptographic methods and audit trails, are needed to prevent hackers from sabotaging auctions and buyers and sellers from cheating or disrupting the auctions. Efficient notification mechanisms to inform bidders of the latest bids are required to scale the auction application to large number of bidders.

We have implemented an auction system that is operational now. It supports the breadth of auction styles, interaction requirements, and other attributes. We present the design for this auction application which implements these auction types and allows the seller to choose any of them and further fine-tune the rules to maximize his business objective. We describe key features of the underlying object, process, and interaction models. In a companion paper we discuss various types of auctions in detail [9]. There we also discuss how auctions relate to other types of commonly used trading models such as brokerages, two party negotiations, and competitive bid-based procurement.

## 2. Requirements for an Internet auction application

An auction application must support the various types of auctions practiced routinely around the world. In this section we first briefly review the different kinds of auctions. Given the unique characteristics of Internet, it is highly likely that other forms of auctions will also emerge. Next we discuss the steps of a complete auction based trading process., and then we discuss some requirements for security and access control. In addition to these requirements, the auction application should integrate with the back end ERP (enterprise resource planning) applications of an organization (or spreadsheets for small business and home businesses) in a straightforward manner. ERP applications are the transactional programs that automate a business's various activities such as procurement, sales, invoicing, payments, human resources, etc.

### Different auction methods:

The commonly used auction types are the open-cry auctions, single and multiple round sealed bid auctions and Dutch auctions. In an open-cry auction, also called an 'English auction', the buyers gather at a common location, physical or virtual, at the pre-specified time. Each buyer can hear the bid submitted by a rival buyer and has a limited time to respond to it with a higher counter-bid. In physical auctions the responses must be received within seconds, while in Cyber auctions it is conceivable that several minutes or hours will be allowed for the response.

In a sealed bid auction the buyers are required to submit their bids by a specified deadline. The auctioneer keeps the bid information secret until the deadline, at which time the bids are evaluated and the winners are declared. Single round sealed bid auctions lack the competitive atmosphere (bidding frenzy) in open cry auctions which encourages the bidders to outbid their rivals. Multiple round sealed bid auctions rectify this situation. In a multiround sealed bid auction there is a deadline for each round of bids, and at that deadline either the auction is closed or the bids from the current round are publicized and a fresh round of bids is solicited by some new deadline.

Dutch auctions are better suited for perishable items such as vegetables or airplane seats. Here the auctioneer starts with a very high asking price. Then he gradually decreases his asking price until buyers emerge with bids specifying how many items they will purchase at the current asking price. He can continue lowering his bid to maintain a stream of buyers while the inventory lasts. Furthermore, he can control how fast he depletes his inventory by controlling the rate at which he lowers the bid.

Each of these auction methods has subtle variations such as:

- Anonymity, i.e., what information is revealed during the auction and after the auction closes. For example, the identity of the bidders could be concealed. In a sealed bid auction the final winning prices could be kept confidential. In all auctions the amount of inventory may or may not be announced in advance.

- Rules for ending Dutch and open cry auctions. Open cry auctions may end at a posted closing time. Alternatively the auctions could be kept open so long as new bids continue to arrive within some time interval of the preceding bid. This interval would be several minutes in an Internet auction and a few seconds for an auction being conducted in a meeting room. One could also choose to close the auction if either of the above two conditions is met or only when both conditions are met. Dutch auctions could close at a pre-specified time, when all the inventory has been sold, when the price has fallen to a pre-specified level, or at some combination of these three conditions.

- Once the bidding phase is over, the bidders with the highest bids get the item being auctioned, but the price they pay could be the same as what they bid or lower. In a *Discriminative Auction*, also known as *Yankee Auction*, the winners pay what they bid. In a *non discriminative* auction people with winning bids pay the price paid by the winning bidder with lowest bid. Finally, in an auction for a single item, in a *Vickrey Auction* [10] the winner pays the price bid by the second highest bidder. Vickrey auctions are also referred to as second price sealed bid auctions.

- Restrictions on bid amount: In all auctions the seller can specify the minimum starting bid. To speed up the bidding process minimum bid increments are often enforced. The bid increment is roughly proportional to the current bid, i.e., they are smaller for lower bids and larger at higher bids. The seller may also be allowed to specify a reserve price, which is a lower limit on price acceptable to seller. The buyers may know that a reserve price exists but they may not know what the reserve price is.

Though counterintuitive, Vickrey, Dutch, open cry and sealed bid auctions yield the same revenue for the seller when one item is being sold, the bidders are risk neutral, their valuations are independent draws from the same probability distribution, and each bidder knows only his valuation and the distribution function from which the other bidders draw their valuation. The result is known in literature as the *Revenue-Equivalence Theorem*, and the assumptions about the valuation are known as the *independent private value model*. Another interesting property of Vickrey auctions is that they are incentive compatible, i.e., it is in the best interest of the bidder to bid their true valuations.

However, experimental observations of the outcomes of various types of auctions differ from the strategic behavior predicted by the game theory based analysis because of psychological factors such as preference reversal and misjudged revision of valuation and winning probability during an auction [11]. Though the Dutch and first-price sealed bid auctions are strategically equivalent, Dutch auctions fetch a lower price. The Dutch auction price is lower than that predicted by the strategic behavior in traditional analysis. The two plausible explanations presented in [11] are that: 1) bidders derive a positive utility from suspense (they play against the odds to remain at the gambling table); 2) They update (lower) their estimate of other bidder's valuations as the offering price moves down. Similarly, though the Vickrey and English auctions are strategically equivalent, Vickrey auctions fetch a price higher than the English auction. The Vickrey auction bidders bid above their true valuations. The plausible explanation presented in [11] is that the bidder's mistakenly believe that by bidding marginally above the true valuation they increase their chances of winning without risking negative payoff.

In this paper we focus on auctions of single or multiple copies of indivisible goods where the goods being sold come from a single seller, or are aggregated before the auction process begins and are sold with the same set of rules such as reservation or asking price. Beam et.al. address the problem of scheduling auctions for periodically arriving goods where arrival of bids from buyers follows a Poisson process [12]. Many market mechanisms exist to match multiple sellers directly with multiple buyers. Various types of stock exchanges, secondary markets for financial instruments, and commodity exchanges fall in this category. Detailed discussion of these is beyond the scope of this paper. Some interesting examples are the reverse unilateral auctions offered by CXN *(www.cxn.com)* and also discussed by Beam and Fusz in [13], Generalized Vickrey Auctions [14], and continuous double auctions [15]. In the CXN auctions all sellers specify their asking price and the auctioneer communicates the lowest asking price to the potential buyers. In a Walrasian auction each bidder specifies a bid schedule consisting of multiple price points and the quantity demanded at each price [16]. The trading mechanisms of New York and NASDAQ stock exchanges are described in [17].

## Complete auction process:

A complete auction-based trading process comprises six basic activities:

1. **Initial buyer and seller registration:** This step deals with the issues relating to authentication of trading parties, exchange of cryptography keys, and perhaps creation of a profile for each trader that reflects his interest in products of different kinds and possibly his authorized spending limits.

2. **Setting up a particular auction event**: This step deals with describing the item being sold or acquired and setting up the rules of the auction. The auction rules explain the type of auction being conducted (open cry, sealed bid, Dutch), parameters negotiated (price, delivery dates, terms of payment, etc.), starting date and time of the auction, auction closing rules, etc.

3. **Scheduling and advertising:** To attract potential buyers, items of the same category (art, jewelry, rare coins) should be auctioned together at a regular schedule. Popular auctions can be mixed with less popular ones to force people to be present in the less popular auctions. Items to be auctioned in upcoming auctions are advertised, and potential buyers are notified in this step.

4. **Bidding:** The bidding step handles the collection of bids from the buyers and implements the bid control rules of the auction (minimum bid, bid increment, deposits required with bids) and for open cry

auctions notifies the participants when new high bids are submitted.

5. **Evaluation of bids and closing the auction:** This step implements the auction closing rules and notifies the winners and losers of the auction.

6. **Trade settlement**: This final step handles the payment to the seller, the transfer of goods to the buyer, and if the seller is not the auctioneer, payment of fees to the auctioneer and other agents (appraisers, consignment agents, etc.).

### Security Requirements:

The auction house policy and the instructions from the seller dictate whether the auction is accessible to the public at large, to the buyers/sellers registered with the auction services, or only to buyers registered to participate in the current auction. Access control mechanisms are needed to enforce these rules. Security mechanisms are needed to ensure that the site announcing the auction and the auction rules is not sabotaged by an outsider. This includes preventing unauthorized postings and alterations as well as preventing denial of service attacks. A trusted third party service for enforcing access control rules and digital signing of contracts to ensure non-repudiation, is discussed in [18].

Cryptographic mechanisms that prove that a particular auction notice was posted and accessible during a certain time period will be very useful in government auctions. During the bidding phase cryptographic mechanisms are needed to ensure that a bid submitted is not tampered with, or disclosed to other bidders in violation of the auction rules. In open cry auctions spurious bids, injected by the seller or auctioneer to prompt the highest bidder to further increase his bids, must be prevented by establishing a verifiable connection from every bid to a known bidder. In the real world such unethical behavior is called taking bids off the wall, or ceiling. A shill is a human agent deployed to inject spurious bids into an auction.

Franklin and Reiter describe a secure protocol for eliminating fraudulent activity by the auctioneer [19] in a sealed bid auction. It employs multiple auction servers, at least some of which are trusted to stay honest. This certainly is useful to control employee fraud within a large auctioneer's organization. However, for mass trading on the Internet where many small organizations are involved, placing trust in the organization itself is an issue. In simplistic terms, it is reasonable to expect that, say three out of five employees (servers or server administrators) in a government organization or *xyz_megacorp* will be honest. But it is difficult to assume that three out of five servers deployed by the relatively small *xyz_little_corp* will not collude.

Spurious or phantom bids are possible in real auctions because knowledgeable or well known buyers often want to remain anonymous (when Leonardo DeVinci's diary was won by Bill Gates; the buyer's identity was discovered only after the auction was over). The presence of knowledgeable bidders in an auction prompts other bidders to bid high undermining the interest of the knowledgeable bidder. Accommodation of anonymous bidders gives the auctioneer an excuse to pick bids from 'nowhere'. Internet auctions cannot overcome this mechanism design constraint. The problem is further aggravated because Internet auctions will have much larger participation from geographically dispersed bidders and cyber identities can be created easily. The possible solution approaches to this problem are *caveat emptor*, requiring mechanisms to let bidders establish the identity of other bidders, or an independent third party trust rating system which can investigate the ethical behavior of the auctioneer and assign a trustworthiness rating to it. Shills are detectable by experienced bidders because when shills are the winner in an auction, which happens some times accidentally, the items reappear on the auction block. Shills appear more frequently at auctions and tend to bid on unrelated items. However, it is easy to hide these clues on the Internet.

## 3. Design of Auction Software

In this section we describe the design of an auction application. Figure 1 shows the object model and Figure 2 shows the data flow diagram of a generic auction application. It supports the various types of auctions discussed in the previous section.

### Object Model for auction application:

In Figure 1 multiple products, traders, and auctions are part of the auction house. Each auction is for one or multiple copies of a single product and a subset of traders participate in it. The product object describes the product or service being auctioned. It is separate from the auction object because the same product can be auctioned in different quantities in different auctions held at different times. For example, a seller with 500 widgets may wish to sell 100 widgets on each day of a chosen week. The search object supports various search methods on the products in the store. Its attributes will be the different classifications of products required by the search methods.

In addition to products and participants, the auction object comprises messages created by participants, terms of the final sale, rules of the auction, and the final trade generated at the end of the auction. The important message types in an auction are query, create, update, and delete bid messages send by the buyers; and messages sent by the seller to close the auction manually or change the asking price in a Dutch auction.

The *state* attribute of auction restricts the type of messages a participant can create when the auction is in that state. For example a simple open cry auction can be in one of three states, active, closed, or not-started. Changing rules and parameters of the auction, like adding or changing the reserve price or minimum starting bid, would be allowed only by the seller while the auction is in the not-started state. Similarly, buyers may be allowed to submit bids while the auction is in not-started or active state, but they may be allowed withdraw bids only when the auction is in not-started state. The *accept-message* method of *auction* object queries the auction rules external to the auction object to determine if a message can be accepted.

Each message, for example, bid from a buyer or closing of auction by the seller, results in updates or notifications being sent to all or some of the other participants. Different participants may get different notifications. For example, when a bid is submitted in an open-cry auction the bid submitter gets a simple acknowledgment, the bidders whose bids are lower than the newly submitted bid will be informed that a higher bid has been submitted, while bidders whose bids are already higher than the newly submitted bid may get no notification at all. Similarly, when the auction closes, the winners and losers will get different notifications. The notification object encapsulates multiple transport methods to send the notifications to the participants. E-mail, online notification via a push or pull mechanism, and manual polling by the bidder are possible. The rules object provides the method to determine which notification methods must be used with each message.

The rule object contains methods which modify the behavior of other objects. Two examples cited

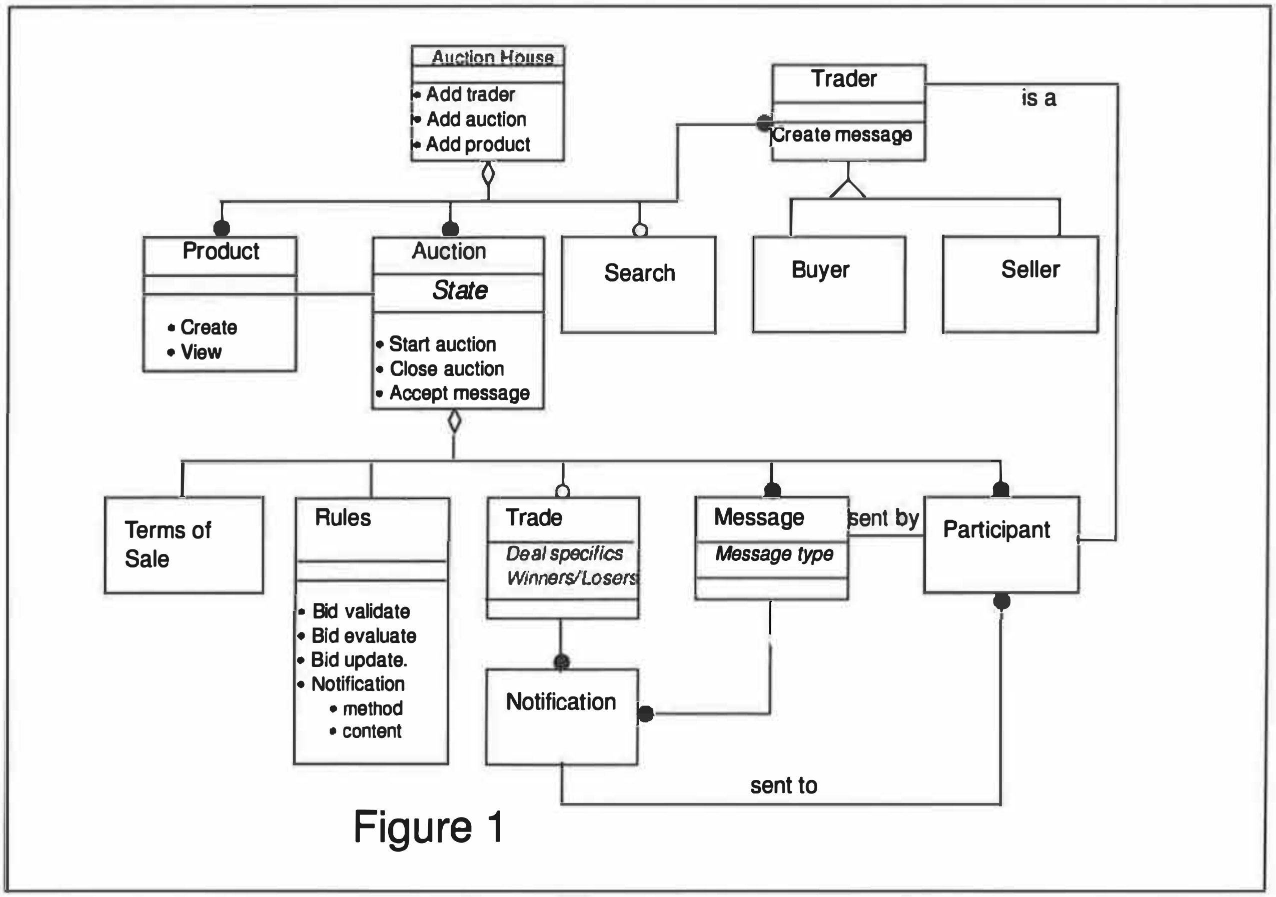


Figure 1

earlier are rules governing when a certain type of message is accepted by the auction object and the rule governing the method used to notify participants. Rules which need to be set or fine-tuned by the auctioneer or seller are specified in the rule object and methods are provided to review and set these rules.

The trader object basically contains the registration information for the buyers and sellers. In addition to name, address etc., it may contain buyer preferences for products, preferred method for being notified of auction updates, and passwords. It also contains a list of auctions on the auction site that a bidder has participated in or has expressed an explicit interest in. We refer to this short list as the bidder's personal auction gallery.

The auction object and the rules object would be subclassed to support different kinds of auctions. For example, the Dutch auction object derived from the basic auction object would provide a method to accept a message from the seller which revises the asking price downwards. Similarly the bid objects can be subclasssed. All bids would share common sending, receiving, and archiving methods, but the Dutch auction bids would not allow a non null value for price (the Dutch auction bid contains amount, the price being quoted by the auctioneer/seller).

## Process flow model:

In the process flow diagram in Figure 2, the three rectangles (buyers, sellers, and settlement system) are the active producers or consumers of information at the boundary of the data flow graph. The disk symbols are information repositories internal to the data flow model. Ovals represent processes, solid arrows represent flow of information, and dashed arrows represent control signals. A hollow triangle at the end of a line indicates a repository created by a process, some of which can be temporary. The five steps of the auction process mentioned in the previous sections are identified in the data flow model at the bottom of Figure 2.

As shown in Figure 2, buyers and sellers can register themselves to create trader objects. Additionally, buyers register their preferences through the preference process and sellers can use this information to create target lists to promote to promote a product or class of product to buyers in a target list. The sellers use the define/update process to create a description of the product being auctioned and use the Setup auction process to describe the type of auction and various rules that go along with it. If the auction rules permit they can cancel the auction or update some of its parameters such as closing time. The alert process notifies a buyer when a product is placed on auction that matches his preference, or if the auction rules or product information changes for such products. The search process provides various search metaphors to help the buyer find/select an auction. The selected auction can either be entered in a list of interesting auctions or the buyer can proceed to bid on it. In the latter case notifications may be sent to bidders whose bids are superseded.

The seller can also select an auction, and in case of Dutch auction use the offer process to change the asking price. This again results in notifications being sent to people who are participating in the Dutch auction (participation in a Dutch auction by buyers who have not bid at all can be recorded by keeping a zero quantity bid in the bid table). Finally the close (auction) process normally closes the auction automatically based on auction rules, but supports manual closing by the seller. When an auction is closed, all the bids submitted for that auction are evaluated and the winners and losers are notified (Dutch auctions, by definition have winners only). Shipping and payment instructions pertaining to each wining bid are sent to the back-end ERP system for final settlement.

Each buyer can maintain a list of auctions that are of interest. This list is maintained at the auction server. The auctions in which a buyer participates are added automatically to the short list. The buyers select auctions from the list or through the search processes provided and submit bids for auctions in progress. In auctions of multiple items, buyers are allowed to submit multiple bids. Depending on the cutoff price for winning bids, and the cutoff time for bids sharing the cutoff price, all, some or none of them could be the winning bids. Currently we do not implement the ability to bid on a bundle of goods. We also do not support the ability to submit a set of bids for the same or different items where only one of the bids, the best one based on the evaluation criterion deployed, will be considered. The buyers can review the bids they have submitted and increase/modify the bids. In open cry and Dutch auctions the buyers can review the bids submitted by other bidders.

The Auction application is fundamentally driven by the auction rules repository. It includes the schedule for the auctions, templates for creating the popular kinds of auctions, and the rules governing individual auctions. Different product in the product repository can be auctioned using different auction types

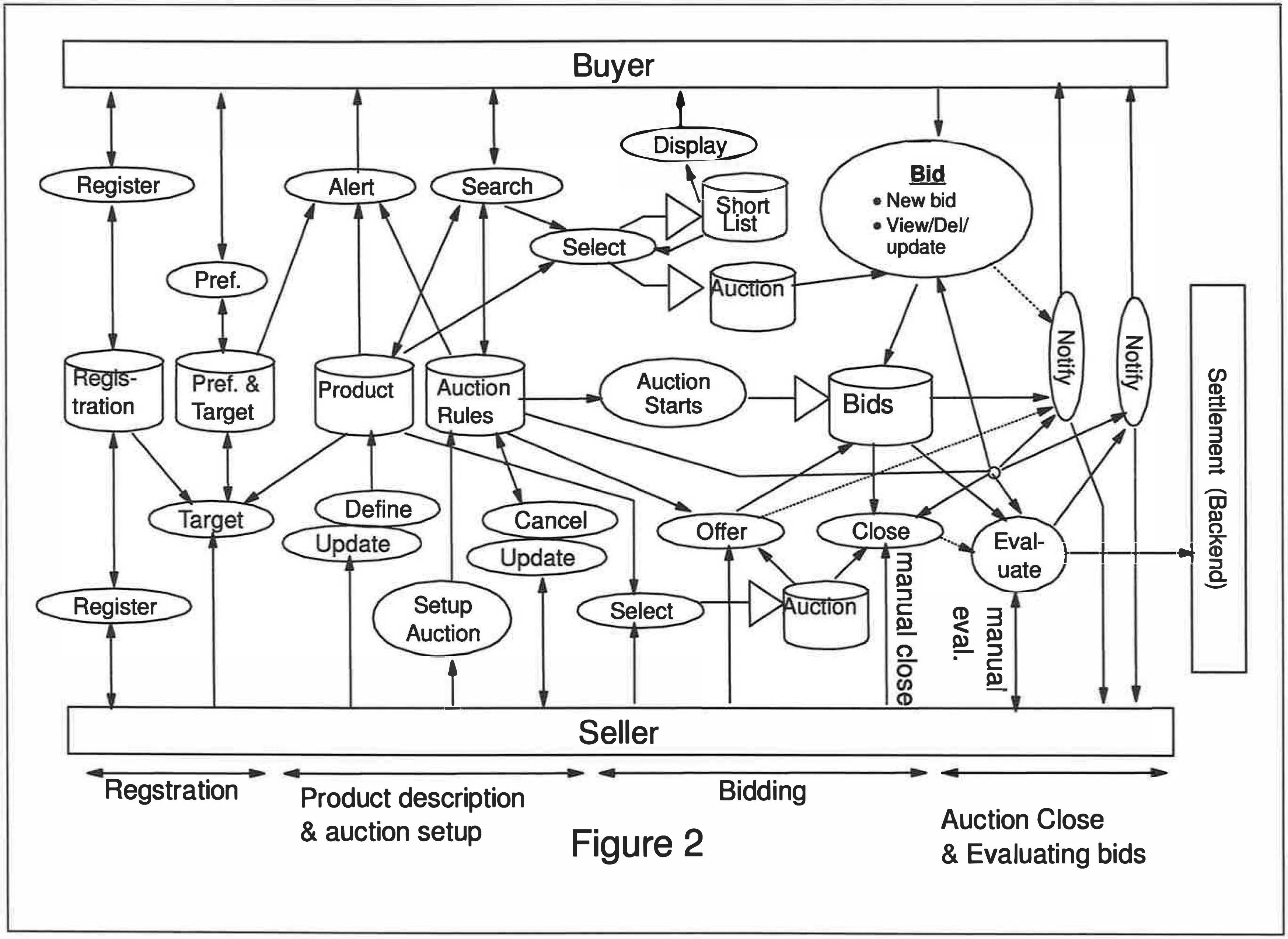

Buyer
Register
Pref.
Alert
Search
Display
Short List
Select
Auction
Bid
• New bid
• View/Del/ update
Regis- tration
Pref. & Target
Product
Auction Rules
Auction Starts
Bids
Notify
Notify
Settlement (Backend)
Target
Define
Update
Cancel
Update
Offer
Close
Eval- uate
Register
Setup Auction
Select
Auction
manual close
manual eval.
Seller
Regstration
Product description & auction setup
Bidding
Auction Close & Evaluating bids

Figure 2

and policies. As shown in Figure 2, the auctions rules define:

**Bidding process:**

1. What is the content of a bid, i.e., price and quantity for a regular auction or quantity only for a Dutch auction.
2. Under what conditions can a previously submitted bid be withdrawn.
3. The minimum bid, bid increments, and deposits required with bids.
4. The information sent back to the buyers and sellers in response to bids received. For example, in an open cry auction the notification to the buyers in response to a bid would be some subset of highest bids, and a subset of information from each bid. For example, the bidder's identity could be dropped. In a sealed bid auction only the bidder sending the bid will get a simple acknowledgment.
5. In open cry and Dutch auctions, which subset of the bidding history is accessible to the bidders.
6. How the notifications are sent back: e-mail, live sockets, etc.
7. Which subset of buyers are eligible to submit bids.
8. Are bidders allowed to submit multiple bids when multiple items are on sale.

**Seller's options:**

1. The seller's ability to modify the sell offer by lowering published price in a Dutch auction, or the inventory being auctioned.
2. Auction closing rules, i.e., whether the auction ends by seller's manual intervention, at a fixed time automatically, after a fixed period of bidder inactivity, or some combination of the three.
3. Under what conditions, if any, can the seller change the auction rules or withdraw the auction.
4. Rules for resolving tied bids, i.e., whether an earlier bid gets the priority or repeat customers are given higher priority.
5. Complex bid evaluation rules like giving weight to bids specifying large product quantities or prompt payment.

## Navigation

Figure 3 shows how the bidders navigate the auction web site. Each bubble shows a web page and arcs from one page to another indicate that a hot link is available from the first page to the second. The seven pages marked with asterisk can be accessed any time from the side bar. The auction site URL puts a bidder on the Welcome page from where a registered user can authenticate himself to the web site and initiate a secure session (login). An unregistered user will get the opportunity to fill in a registration form which may be processed online or off-line. After registering, the bidders can browse through or search the products in the auction house which will possibly result in a product being selected and its description presented to the bidder. If the product is on auction, the rules of auction can be viewed, and bids can be submitted for that product.

From the home page the bidder can also see a list of all auctions at the auction site or a subset of these which are in his personal auction gallery. From either list, the bidder can select an auction and access the description of the product being auctioned, see the rules of the auction, or bid on the product. For open-cry auctions he can also see a subset of the previously submitted high bids for that product. In both these lists, the entry for an auction also includes the auction type, quantity being auctioned (if the rules permit), best bid or current asking price for open cry and Dutch auctions respectively, and auction closing time if determined.

From the list of high bids for a product mentioned above, a buyer can increase his existing bid, submit a new bid, review the auction rules or access the product description. Similarly, from the home page a buyer can access the list of all his bids and perform the same functions. Finally from the home page a buyer can access his message box which contains all the notifications sent by the auctioneer to the buyer. When ever there is an unread message in the mailbox of the buyer, an icon flashes in the side bar on all web pages. This icon is in a small frame which refreshes itself every few seconds (Netscape client pull feature. Refresh time is specified in the HTML header).

## Notification:

Currently two notification mechanisms are provided in the auction prototype. First one is the simple e-mail. The second one is through the message box mentioned in the previous paragraph. E-mail is necessary to communicate with buyers who are not looking at one of the auction site pages when a message need to be delivered. Message box is more convenient for those who are on some auction site page when the message is to be delivered.

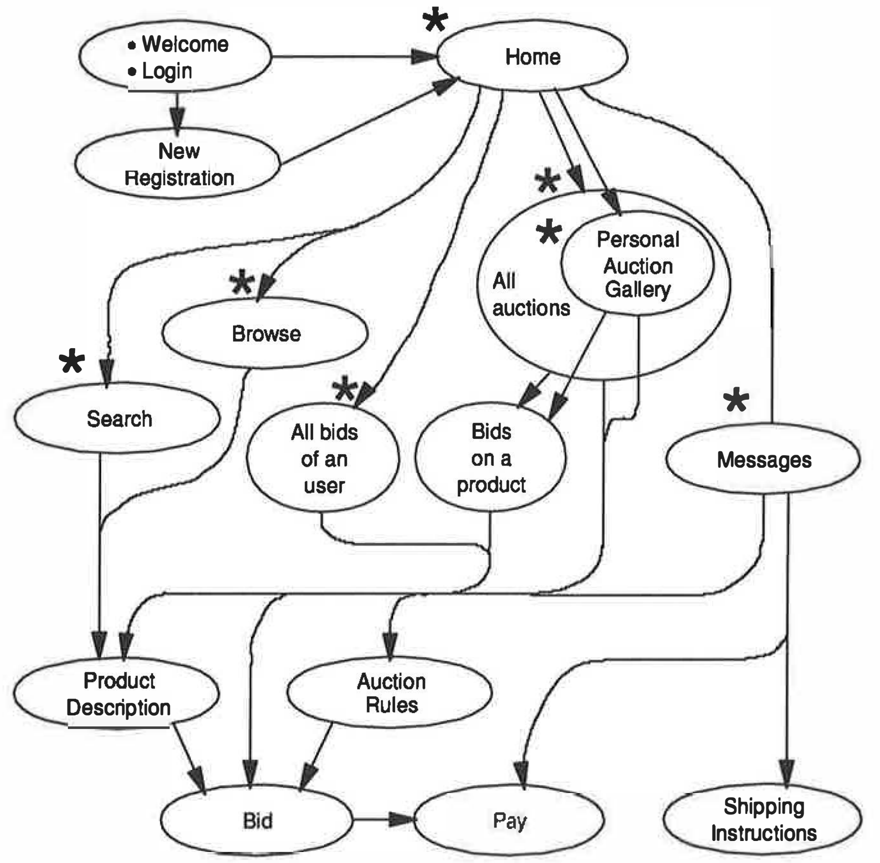


Figure 3

Other than the mailbox icon in the side bar mentioned in the previous page, only the bid submission page is updated dynamically to show the current highest bid or current asking price for open cry and Dutch auctions respectively. This update is also accomplished by putting the bid/asking-price information in a separate small frame and specifying a short refresh time for it. Because the page refresh mechanism is a client side polling mechanism, it is not very scalable. Most of the time the server will be repeating its old information to the client. In a future implementation we plan to use server-side push to a java applet sitting on the client browser. On the server side an optimized notification process would update messages only to clients which require the update. Most of the time an identical update will be sent to multiple clients. The notification process would allow a message to be sent repeatedly to different clients by stepping through the socket addresses (or an equivalent network address entity) in the inner loop of the code.

## 4. Internet-specific design issues

Auctions in the real world are well understood and studied extensively. However the Internet changes the auction environment, mitigating some constraints of the real world and introducing some new problems of its own. In this section we discuss how auctioning on the Internet could differ from auctioning in the real world.

### Bidder collusion:

A set of bidders in an auction can collude to form a ring, where the members of the ring agree not to outbid each other. At the end of the auction, if the item is won by a ring member, it is resold among the ring members using a separate auction, or some other allocation procedure. The surplus created in the second sale is the loss inflicted to the seller. It is split among the ring members. Internet makes the fonnation of rings much easier. Just like hacker's web sites and chat rooms, one can conceive of bidder's chat rooms with cyptographically secure mechanisms for creation of rings. Even in the absence of such centralized chat rooms, if the identities of bidders are known to other bidders, a desirable situation to reduce phantom bids by the seller, the formation of rings is facilitated. Aggressive use of reserve prices can reduce the incentive to form rings by reducing the gains expected by their creation.

Luckily, rings are illegal, though not necessarily unheard off. Under commerce laws disruption of a fair market process is illegal and forming rings amounts to disrupting a fair market process. One can expect some debate on applicability of today's outmoded commerce laws to the cyber world, but history indicates that legal apparatus will move to protect the interest of commerce.

### Choosing the right kind of auction:

There are reasons for choosing one fonn of auction over other. Vickrey auctions eliminate the effort on the part of the bidder to speculate on the minimum bid he needs to submit. Dutch auctions provide the seller a better control on (liquidating) inventory by giving him the controls to revise the prices downwards and thereby manage his inventory level more directly. Dutch auctions also discourage the fonnation of rings because in a Dutch auction a ring member stands to gain by defecting from the ring while the auction is in progress and the rest of the ring is playing by its rules. Open cry auctions do not encourage defection because the defecting ring member does not increase his gains by defecting. After the defection of one (or few) ring members the rest of the ring can continue to operate as if the defecting ring members were not part of the ring to begin with.

Choice of auction method also depends on the industry in which it is being used. Governments (democracies) are likely to lean towards auction methods which have higher transparency. Sealed bid auctions are likely to be used when preparing the bid is time consuming, or it is impractical for the bidders to collect at a common location at the same time. Since the Internet frees the bidders from this constraint, open cry auctions are likely to be preferred on the Internet (with bidder anonymity if needed).

In practice, open cry auctions are usually for one item. If multiple items are to be sold, they are sold one at a time. That is acceptable in the real world because each item sells fast, and it is impractical to take multiple bids for the items simultaneously. On the Internet one can sell multiple items simultaneously. This is also necessary to some degree because each auction takes a longer time. Therefore, we expect to see an increase in use of auctions for multiple items.

### Withdrawing bids in an open cry auction:

In a traditional open cry auction, which lasts usually for minutes, the bids submitted by a buyer are binding, i.e., he simply can not back of from his bid. Since open cry cyber auctions can take hours or days to conclude, the potential bidders will be hesitant to make such an open-ended commitment to buy. Hence, the Internet open cry auction mechanisms must give the bidder an opportunity to ask the seller for a commitment or withdraw his bid. Decision support tools would be needed on the seller's side to help him decide whether to commit to the sale in this situation.

### Usability:

Traditionally the buyers and the sellers in an auction have been seasoned professionals of an industry with intimate knowledge of the auction mechanism and of the relevant bidding strategies. However, Internet brings auctioning to the masses, and a typical participant may know very little about the often complex auction mechanisms. Thus the usability issues in the design of auction application are extremely important. Not only should the navigation within the application be simple and intuitive, as discussed in the previous section, help should be available on the finger tips on:

- How to use the application software
- Explanation of the auction mechanism deployed
- Bidding options available to the buyer and strategic implications of each option
- For sellers the auction mechanisms available and the implications of choosing one or the other.

### Bidding agents:

An obvious area of extension to electronic auctions is buying agents, which would be programs that can search for auction sites of interest to a buyer and automatically bid on his behalf. These programs could also potentially search the Internet for the final sales prices for a particular product in recently closed auctions and base their bidding strategy on these prices and trends in them.

When both software agents and humans are bidding in an open cry auction it would be desirable to ensure that software agents have similar response time in submitting a new bid as humans. This is based on the belief that bidding is not totally rational and people develop an emotional tie with the product after participating in the bidding phenomenon for a while. If agents were allowed to bid with their millisecond response time, the bids would reach a level very quickly where the humans may be disinclined start bidding.

## 5. Summary and future research

We have implemented an auction application which closely follows the design outlined above and are looking forward to field trials. We are currently exploring practical ways of meeting the security requirements in section 2. Additional tools are needed to make auctions attractive to real business users. They include mechanisms to archive closed auctions for record keeping, support for electronic bidding proxies (order bids), and integration with back-end ERP systems.

Strategic behavior by bidders and equilibria in various auction methods has been studied extensively in economic literature. However, Internet auctions are intended for ordinary people who are not fond of rational or highly strategic behavior. Otherwise, people would not play slot machines or lottery, and they would not stand in long lines to get a ticket for the first show of a blockbuster movie. It is important to investigate the psychological and emotional aspects of behavior of people in an auction and factor the results in the design of auctions.

## References:

[1] H.G. Lee, "Do Electronic Market Marketplaces Lower the Price of Goods," Comm. of the ACM, Vol. 41 No. 1, Jan. 1998, pp. 73-80.

[2] H.G. Lee and T.H. Clark, "Impact of the Electronic Marketplace on Transaction Cost and Market Structure," International Journal Of Electronic Commerce, Vol. 1 No. 1, Fall 1996, pp. 127-149.

[3] Efraim Turban, "Auctions and Bidding on the Internet: An Assessment," International Journal of Electronic Markets, Vol. 7 No. 4, http://www.electronicmarkets.org/

[4] Ralph Cassady Jr., "Auctions and Auctioneering," Reading, Univ. California Press, Ont. 1979, ISBN 0520002164.

[5] R.P McAfee and John McMillan, "Auctions and Bidding," Journal of Economic Literature, Vol. 25 No.2, June 1987, pp. 699-738.

[6] Paul Milgrom, "Auctions and Bidding: A Primer," Journal of Economic Perspectives, Vol. 3 No. 3, Summer 1989, pp. 3-22.

[7] Paul Milgrom and R.J. Weber, "A Theory of Auctions and Competitive Bidding," Econometrica Vol. 50 No. 5, Sept. 1982, pp. 1089-1122.

[8] J.H. Kagel and A.E Roth, "The Handbook of Experimental Economics," editors J.H. Kagel and A.E. Roth, Princeton University Press 1995, ISBN 0-691-04290-X.

[9] M. Kumar and S.I . Feldman, "Business Negotiations on the Internet," proc. inet'98, Geneva Switzerland, July 21-23 1998.

[10] David Vickrey, "Counter Speculation, Auctions, and Competitive Sealed Tenders," The Journal of Finance, March 1961, pp. 9-37.

[11] J.H. Kagel, "Auctions: A Survey of Experimental Research," in reference [8] above.

[12] Cary Beam, Arie Segev, an J.G. Shanthikumar, "Electronic Negotiations through Internet-based Auctions," CITM Working Paper 96-WP-1019, Dec. 1996, Walter A. Haas School of Business, Univ. of California Berkeley, Berkeley CA 94720.

[13] Cary Beam and Gene Fusz, "CXN: A Case study," CITM Working Paper 97-WP-1025, Oct. 1997, Walter A. Haas School of Business, Univ. of California Berkeley, Berkeley CA 94720.

[14] Hal R. Varian, "Economic Mechanism Design for Computerized Agents," First USENIX Workshop on Electronic Commerce, July 11-12 1995, pp. 13-21.

[15] P.R. Wurman, W.E. Walsh, and M.P. Wellman, "Flexible double auctions for electronic commerce:

Theory and implementation," to appear in Decision Support Systems.

[16] Tracy Mullen and M.P. Wellman, "Market-Based Negotiation for Digital Library Services," Second USENIX Workshop on Electronic Commerce, Nov. 18-21, 1999, pp. 259-269.

[17] Steven M.H. Wallman, ""Technology takes to securities trading," IEEE Spectrum, Feb. 1997, pp. 60-65.

[18] P. Sanders, S. Rhodes, and A. Patel, "Auctioning by Satellite using Trusted Third Party Security Services," 11th IFIP conference on information security, 1995, pp. 205-219.

[19] M.K. Franklin and M.K. Reiter, "The design and implementation of a secure auction service," IEEE Trans. on Software Engg. 22(5), May 1996, pp. 302-312.

# Electronic Auctions with Private Bids

Michael Harkavy
*Carnegie Mellon University*
*Pittsburgh, PA 15213*
bif@cs.cmu.edu

J. D. Tygar
*Carnegie Mellon University*
*Pittsburgh, PA 15213*
tygar@cs.cmu.edu

Hiroaki Kikuchi
*Tokai University*
*Hiratsuka, Kanagawa, Japan*
kikn@ep.u-tokai.ac.jp

## Abstract

*Auctions are a fundamental electronic commerce technology. We describe a set of protocols for performing sealed-bid electronic auctions which preserve the privacy of the submitted bids using a form of secure distributed computation. Bids are never revealed to any party, even after the auction is completed. Both first-price and second-price (Vickrey) auctions are supported, and the computational costs of the methods are low enough to allow their use in many real-world auction situations.*

## 1 Introduction

Auctions are a fundamental technology for electronic commerce. They have been suggested as a technology for controlling allocation of bandwidth [5, 10] and are increasingly seen on the web.

If we could have an ideal auction, what properties might we desire? Here are some desiderata (not a complete list) for an ideal auction:

- **Economic design** — we want the auction to be designed on solid economic principles and for participants to have incentives to bid as they truly value the item — this is known as their *valuation*, and is also called their *indfference price*. If bidders bid less than their true valuations, it is possible that the final winning bid may be artificially low — we illustrate this below in our discussion of sealed-bid auctions.
- **Fast execution** — we want to have the auction run quickly.
- **Privacy** — we want the auction to be private — for others to not know our actual bids. We do not want even an auctioneer to know the bids. The only exception to this rule is that we will reveal the final price at which the item is sold. (At first this may seem like a paradoxical condition, but it is commonly achieved in Dutch auctions, discussed below.) Note that this is a quite useful requirement — otherwise we give away detailed information on our preferences that may be used in the future to inform "shills" who work for the seller to attempt to artificially drive up the price an item is sold at (creating a disincentive to bid the true valuation.)
- **Anonymity** — we don't want our identities to be revealed. One way to achieve this is to use an intermediary to anonymously forward our bids. Note that privacy is different from anonymity; privacy protects the values of the bids while anonymity protects the identities of the bidders. Even if our bids are anonymously forwarded, participants (such as the auctioneer) may learn the distribution of our bids.

In this paper, we discuss how to hold a true auction that combines the first three features. If we add anonymizing intermediaries to the mix, we can achieve an auction with all four properties.

### 1.1 Auction Types

How do existing auction types stack up against our desiderata?

Consider these three broad categories of auctions that have been proposed:

- *Increasing-price auction (English auction).* In this type of auction, a good or commodity is offered at increasing prices. It may initially be offered at $K$ tokens; at successive points of time $i$ it is bid at $K + i * \Delta$ tokens ($\Delta$ may be a function of previous bids and other factors). At each unit of time, one or more parties

As of September 1, 1998, the first two authors can be reached at Computer Science Division, University of California, Berkeley, CA 94720. Sponsored by DARPA under grant F19628-96-C-0061, the U.S. Postal Service, and Toshiba Corporation. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation thereon. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official poilicies, either express or implied, of any of the supporting organizations or the U.S Government.

can bid for the item. At the end of the auction, the highest bidder takes the item; he pays the price he bid. This is the sort of auction found at Sotheby's and Christie's. This type of auction has many disadvantages: the time necessary to conduct the auction is potentially proportional to the price at which the item is sold; the communication costs may grow superlinearly in the ultimate price at which the item is sold (since at lower prices, multiple bidders may simultaneously bid for an item); moreover, this type of auction leaks an enormous amount of information—a careful observer will be able to deduce information about the price that each party is willing to pay for the auctioned good. However, the auction does have a very desirable feature: in economic terms, it allocates the good to the bidder with the highest valuation, since the bidder with the highest valuation will be willing to outbid all other bidders.

- *Sealed-bid auctions.* In this type of auction, each party sends a sealed bid to an auctioneer who opens all bids. The auctioneer determines the highest bid and sells the item to that bidder for the bidding price. This type of auction can execute in a single round of communication between the bidders and the auctioneers. However, it has disadvantages. First, the auctioneer will know the exact price that each party is willing to pay. Second, it does not support optimal distribution of goods.

  In a sealed bid auction, participants will have beliefs about what others will bid. If a participant believes that she will have the highest bid, and the second highest bid will be substantially beneath that, then she has an incentive to lower her bid. For example, if she values an item at $1,000, but believes that the second highest bidder values the item at $500, then she is likely to place a bid slightly higher than $500. If she is wrong about the distribution of other bids, then the final item will not go to the party that values it most, and the seller will have given up the item a price lower than he would have achieved with an English auction. [9, 11, 13, 14].

- *Decreasing-price auction (Dutch auction).* This type of auction is similar to the English auction in that the bidding price varies over time; however, in this case, the price decreases and at time $i$ is $K - i * \Delta$. The first bidder will take the item. This type of auction has the advantage of preserving maximum privacy — no information is revealed except the winning bid and bidder; however like the increasing-price auction, it may be time consuming, and like the sealed-bid auction, it is not economically efficient [9, 11, 13, 14].

In Nobel-prize winning work, the economist Vickrey designed a type of auction that combined the best features of an increasing-price auction and a sealed-bid auction [13]. Vickrey's technique, called a *second-price auction*, works like a sealed-bid auction, in that all bids are sealed and sent to an auctioneer. Like a sealed bid auction, the highest bidder wins. But the price the winner pays is the price that the second highest bidder has bid. For example, suppose that we bid 100 tokens and the second highest bid is 10 tokens. Then we will win the bid, but we will only have to pay 10 tokens to secure the good. This auction runs in constant time, and maximizes consumer surplus, but it is still highly centralized and does not protect the privacy of the bids.

The main contribution of this paper is to give a private-bid version of a second-price auction. This auction

- will run in a single round of bid submissions (like a sealed-bid auction),
- is efficient enough for practical implementation,
- will maximize consumer surplus and will give incentives for participants to submit bids at their true valuations (like an English auction), and
- will preserve bid privacy (like a Dutch auction).

This is quite an unusual result. In the end, only the second highest bid is revealed — the auctioneers and participants (except for the winner) will be completely unaware of the numerical value of the highest bid (or any other bid besides the second highest).

## 1.2 Secret computation

Our approach builds on a long tradition of secret function computations. Researchers have developed a number of techniques for securely computing arbitrary functions—some major examples are [2, 8, 3]. This means that given a set of participants, each with an input, we can compute (with a polynomial slowdown) any function of the inputs; and moreover, we can do so with a protocol that leaks no information to any participant. The only information that a participant will know about the the input of other particpants is information derivable from his own input and the final result.

Unfortunately, the general techniques for secure computation are impractical for general use. They do, in fact, run in polynomial time, but with a big explosion of states. Thus, the development of private protocols for specific problems must be done by hand. We have developed and tuned our protocol to run quickly — and we believe that it is practical for real use. We plan to build a prototype system embedding this algorithm to prove its practicality.

In the remaining sections, we first consider a simple private sealed-bid auction; that is, we show how to compute the *max* function privately. Then, we describe a fully private and secure second-price auction protocol which is the result of several improvements on the original.

The essence of the second-price protocol is a series of computations which securely determine whether or not there are at least two bids greater than or equal to a specific value. This test is performed by determining if there is some partition of the set $\mathcal{B}$ of $n$ bidders into $\mathcal{B}_1$ and $\mathcal{B}_2$ such that there is a bid at least as high as the test value on each side of the partition. We need not actually consider all possible partitions; we can select a small number ($\log_2 n$) of partitions which will suffice. We iterate this series of computations in a search for the value of the second highest bid.

## 2 A simple auction method

First, we will briefly describe a simple first-price auction protocol with private bids which captures some of the flavor of the more complex version to follow. We assume that all bids are drawn from some ordered set $\Delta = \{\delta_1, \ldots, \delta_V\}$, and that there are $n$ bidders and $m$ auctioneers. We also fix a sufficiently large (64–128 bit) prime number $p$. All arithmetic will be computed modulo $p$. Each bidder composes her bid by creating $V$ lists of $m$ integers modulo $p$. The rule for creating these sets is that:

- If the bidder is willing to pay $\delta_l$ for the object, then the $l^{th}$ list of numbers is chosen randomly with uniform distribution subject to the constraint that the sum (modulo $p$) of all numbers must be non-zero.

- If the bidder is unwilling to pay $\delta_l$ for the object, then the $l^{th}$ list of numbers is chosen randomly with uniform distribution subject to the constraint that the sum (modulo $p$) of all numbers must be zero.

Each auctioneer is sent $V$ numbers by each bidder; the $i^{th}$ auctioneer is sent the $i^{th}$ number from each list. The submissions are signed by the bidders, and signed receipts are issued by the auctioneers. The signature by the bidder should be a signature on a hash of hashes of each of the individual values in the list so that the signature can be used to prove a bid at one particular value without revealing the bids at any other values. Strange or non-random bidding behaviors are possible (e.g. being willing to pay $\delta_j$ but not $\delta_i$ for $i < j$). Such inconsistencies could be easily eliminated by methods similar to those outlined in section 4.1.1.

To determine the winning bid, the auctioneers compute, for each $\delta_l$, the sum of the $m$ (one from each bidder) numbers received for that $\delta_l$. To determine whether a particular $\delta_l$ is above or below the winning bid, the auctioneers commit to and reveal their sums for that $\delta_l$, and then compute a sum of all these values (which is equal to the sum of all numbers from all bidders for $\delta_l$). The sum of all values for a particular $\delta_l$ is non-zero (with high probability) exactly when at least one bidder is willing to pay $\delta_l$. The auctioneers find the highest $l$ for which some bidder is willing to pay $\delta_l$, possibly in several stages with a branching search for the highest bid. Once the highest bid is known, the auctioneers can then use the signatures on the bid submissions to prove which bidder is the winner by reassembling all bids for that winning $\delta_{l_0}$.

This auction has some weaknesses. Most significantly, a coalition of an auctioneer and the highest bidder might uncover some information about the second highest bid. This information would be leaked by the fact that, for all $l$ lower than the highest bid but higher than the second highest bid, the total of all bids would be simply the bid of the highest bidder. The obvious way to avoid revealing these intermediate sums would be to reveal sums one at a time from $\delta_V$ down and would require a number of rounds linear in the number of bidding points, which is likely to be unacceptably slow. Another weakness is that the work required (both computation and communication) scales linearly with the number of bidding points. For high value auctions, the desirable number of bidding points might be quite large.

We will give an auction in which the work scales only logarithmically with the number of bidding points, which provides complete privacy, and which allows for second-price auctions. It is worth noting that these alterations need not be combined; any subset may be implemented and computation costs will vary accordingly.

## 3 Secure distributed computation

This section provides an overview of the methods used as the computational basis for the protocol, which come from Ben-Or, Goldwasser, and Wigderson[2]. For further details or proofs, we refer the reader to this work and its extension by Beaver[1]. Information is distributed among the agents (the auctioneers in our case) by means of polynomials over a finite field as introduced by Shamir[12], but with verifiability as in [2, 4]. These polynomials are stored in a point-value representation, where each of the $m$ agents has a specific point $\alpha_i \neq 0$ in the field at which he knows the value of all shared polynomials. The value of a polynomial at a specific point is known as a share. The initial polynomials will have some fixed degree-$t$. Since higher-order coefficients for these polynomials will be randomly selected from a set including zero, it might be more precise to say "degree bound $t$", but this distinction is never relevant to our results and we will freely ignore it.

Individual pieces of data are represented as points in the finite field and encoded in a shared polynomial as the free coefficient of that polynomial (which is the same as the evaluation of the polynomial at the point 0). All other coefficients are chosen from the field uniformly at random. Over a finite field (e.g. $Z_p$ for prime $p$), any set of $t$ known values of a degree-$t$ polynomial (chosen as described) gives information-theoretically zero information about the polynomial's value at any other point. This property that any $t$ shares of a secret yield no information about the secret's value is known as $t$-privacy. There is a (simple and short) interactive protocol which ensures that the set of shares held by correct agents represents a degree-$t$ polynomial.

The data used in the computation needs to be not only private, but tamper-proof as well. This property is supported by a special selection of the evaluation points. The $\alpha_i$ are chosen to be the set of powers of a primitive $m^{th}$ root of unity (say $\omega$). That $\omega$ is a primitive $m^{th}$ root means $\omega^m = 1$ and for $i : 1 \leq i < m$, $\omega^i \neq 1$. Obviously we must choose a finite field which has such a primitive $m^{th}$ root, but this is not difficult ($m$ is small). Given this choice of $\alpha_i$, the shares of the polynomial are an error correcting code. Up to $\lfloor \frac{m-1}{3} \rfloor$ shares may be missing or incorrect and recovery of the free coefficient will still be possible. The resistance to up to $t$ false values is known as $t$-resilience.

The basic computational operations are the addition or multiplication of two polynomials. The value of a sum or product of two polynomials at a point is the sum or product of the values of the two polynomials at that point. So, these operations are done pointwise, with each agent simply computing the addition or multiplication of the values at the point held. For addition (or multiplication by a scalar), this process is complete, privacy is not interfered with, and no other work needs to be performed. Extra work is required for multiplication.

The difficulty with multiplication is that the degree of the product of two polynomials is the sum of their degrees. The number of points needed to specify a degree-$t$ polynomial is $t+1$, while to specify the product of two degree-$t$ polynomials requires $2t + 1$ points. Repeated multiplications will continue to increase the degree of the shared polynomials. Since the polynomials are shared among a fixed number of agents, the degree of the polynomial will rapidly exceed the ability of the agents to represent it. Repeated multiplication requires some means of reducing the degree of a polynomial without changing or revealing information about its free coefficient. A related difficulty is that the product of two polynomials does not have the randomness property which we required to enforce privacy. Many polynomials of degree $2t$ can not be expressed as the product of two degree-$t$ polynomials (e.g. irreducible polynomials).

These two difficulties are solved together by a stage of communication between the agents which produces a degree-$t$ polynomial whose free coefficient is the same as the free coefficient of the product of 2 shared degree-$t$ polynomials and whose remaining coefficients are uniformly random.

The key observation (which we state without proof) is as follows. If we interpret the $m$ shares of a polynomial as a vector, there is a constant $m \times m$ matrix (based on the the $\alpha_i$) which transforms the shares of any degree-$2t$ polynomial into shares of a degree-$t$ polynomial while leaving the $t + 1$ lowest-order coefficients unchanged. Each agent can compute a vector of values (call it $V_i$) by multiplying its share of the polynomial by a vector of constants (corresponding to row $i$ of the matrix). Now $\sum_i V_i$ is the vector of shares of the transformed polynomial. In order to mask non-uniformity of the coefficients of the product of polynomials, a random degree-$t$ polynomials is generated and added to the vector components before they are shared. The agents sum all received values to compute their shares of the new polynomial, which has the claimed properties.

In this method as described, multiplication can violate our $t$-resilience because false behavior on the part of one of the agents can lead to erroneous values at many others through the communication of the degree reduction. A further level of complication is required to ensure that the method described is in fact carried out correctly, but we will not go into that level of detail here (as before we refer the reader to [2, 1]). Intuitively, exchanges among the agents are used to prove that the agents follow the prescribed rules in conducting the transfer of information during the degree reduction phases. In order to simultaneously maintain $t$-resilience and $t$-privacy, we must be able to tolerate up to $t$ incorrect shares at the beginning of a degree reduction (when the degree of the polynomials will be $2t$. So, we restrict $t$ by $t < \lfloor \frac{m-1}{3} \rfloor$.

# 4 A scalable secret-bid second-price auction

We describe a second-price sealed-bid auction protocol which has the potential to be $t$-private and $t$-resilient, and which is efficient enough for use in large auctions. All polynomials used during the protocol are secret sharing polynomials from $Z_p[x]$ (the set of polynomials over the field of integers less than a prime $p$). The prime $p$ should be chosen to support the assumption that the sum of several random non-zero elements from $Z_p$ is very unlikely to be zero (i.e. that $p^{-1}$ is very small). They are stored in a point-value representation, with each of the $m$ auctioneers (who are indexed by $i$) holding the value at one point. Let $\alpha_i \in Z_p$ be the point held by the $i^{th}$ auctioneer. These polynomials are manipulated by means of their evaluations at these $m$ points.

Each of the $n$ bidders (indexed by $j$) computes a set of secret sharing polynomials whose secrets encode her bid. The bidders distribute the shares of these polynomials among the auctioneers. The auctioneers then perform a multi-round computation to determine the selling price. The winning bidder is then determined by combining certain information from the auctioneers.

## 4.1 Bid submission

Each bidder selects a bid $b_j \in \{0, \ldots, V-1\}$ which is represented in some fixed base $c$ as $b_j = b_{j1} \ldots b_{jd}$ (where $d = \lceil \log_c V \rceil$). This base $c$ representation of the bid is then encoded with $d$ ordered sets of $(c-1)$ secret-sharing polynomials, each of which has degree $t \leq \lfloor \frac{m-1}{3} \rfloor$. Each set of polynomials encodes one of the digits in a unary style as follows: Let $z \in \{0, \ldots, c-1\}$ be the value of the digit to be represented by a particular set. The first $z$ polynomials are chosen such that their free coefficients are uniformly randomly selected from $\{1, \ldots, p-1\}$. The remaining $c - z$ polynomials are chosen with their free coefficients set to 0. We refer to the $l^{th}$ polynomial of the set which encodes the $k^{th}$ digit of the $j^{th}$ bidder as $s_{jkl}$. We refer to the evaluation of this polynomial at the point controlled by the $i^{th}$ auctioneer as $s_{jkl}(\alpha_i)$.

The bidders will use asymmetric keys to provide accountability and to enable the winner to claim her good. In the normal case, these keys will be the bidders' published public keys. (In the case where a bidder wishes to remain anonymous, a pseudonymous key can be used instead. Issues raised by anonymous bidders are addressed in section 4.6.4.)

Let $M_{ij}$ be the string

$$s_{j11}, \ldots, s_{jkl}, \ldots, s_{jd(c-1)}$$

The bid submission messages are of the form

$$B_j \rightarrow A_i \;:\; E_{A_i}[M_{ij}], D_{B_j}[h(M_{ij})].$$

Note that we are ignoring lower level details such as message identifiers. $D_q$ and $E_q$ represent signature and encryption with an asymmetric key pair, and $h$ is a crytographically secure hash function. The submission is verified to be a valid polynomial using the interactive protocol mentioned in section 3.

### 4.1.1 Incorrect bids

The selection method for the polynomials $s_{jkl}$ requires their values at 0 to be either 0 or a uniformly random selection from $Z_p \setminus \{0\}$. This property is needed to ensure a low probability of error. Since $p$ is prime, the property may be enforced by multiplying each $s_{jkl}$ by an independent scalar value uniformly selected from $Z_p \setminus \{0\}$. Since this is multiplication by scalar values, the degree of the polynomials is not altered. Including this step does not eliminate the need for an honest bidder to select her polynomials randomly as described, since a non-random selection could leak information about the bids.

If $c > 2$, then the method of representing bids allows a bidder to submit a bid which does not correspond to a single value. This is due to the unary-style representation of the individual digits. We are unaware of any advantage that could be derived from using such malformed bids, but they could be easily prevented (if desired) as follows. After all bids are received, the auctioneers select a random number $r \in Z_p \setminus \{0\}$. The $s_{jkl}$ should then be recomputed in order of descending $l$ by

$$s_{jkl} \leftarrow s_{jkl} + r \cdot s_{jk(l+1)}.$$

For the base case, $l = c - 1$, the polynomial $s_{jk(c-1)}$ remains unchanged. Since $r$ is a scalar rather than polynomial, all these computations can be performed without any communication (other than agreeing upon $r$).

## 4.2 Bid resolution

The auctioneers use the submitted bids to compute the selling price $b_0$, which will be equal to the second highest bid submitted. This computation is performed over $d$ rounds, one digit per round, going from most significant to least significant. We will use $k$ to refer to the number of the current round. The value of the digits of $b_0$ is not kept secret (from the auctioneers), but is known to all auctioneers as it is computed.

Some shared polynomials $u_{jk}, v_{jkl}, w_{jk}$ will be computed from the $s_{jkl}$ over the course of price determination. Intuitively, these values are used to track the bid $b_j$ of bidder $B_j$ as compared to the first $k-1$ digits of the selling price.

- The value $u_{jk}(0)$ will be non-zero exactly when $\left\lfloor \frac{b_j}{c^{d+1-k}} \right\rfloor > \left\lfloor \frac{b_0}{c^{d+1-k}} \right\rfloor$.
- The value $v_{jkl}(0)$ will be non-zero exactly when $\left\lfloor \frac{b_j}{c^{d-k}} \right\rfloor \geq \left\lfloor \frac{b_0}{c^{d+1-k}} \right\rfloor c + l$.
- The value $w_{jk}(0)$ will be non-zero exactly when $\left\lfloor \frac{b_j}{c^{d+1-k}} \right\rfloor \geq \left\lfloor \frac{b_0}{c^{d+1-k}} \right\rfloor$.

While the comparisons above use the value of $b_0$, the results are determined entirely by the first $k-1$ digits, which are known prior to round $k$. The $u_{jk}$ are indicators for the winning bidder. The value $u_{jk}(0)$ is non-zero when the first $k-1$ digits of the bid $b_j$ indicate that the bid is greater than $b_0$. Since the selling price is the second highest bid, $u_{jk}(0) > 0$ means that bidder $j$ has the highest bid. The $w_{jk}$ are indicators for the losing bidders. The value $w_{jk}(0)$ will equal 0 when the first $k-1$ digits of the bid $b_j$ indicate that the bid is less than $b_0$. So, $w_{jk}(0) = 0$ means that $b_j$ is at most the third highest bid and that the particular bid $b_j$ will have no effect on the outcome of the bidding.

The $v_{jkl}$, which are computed in round $k$ by $v_{jkl} = u_{jk} + s_{jkl} \cdot w_{jk}$, indicate whether a bid is at least as high as some specific test value. The value $v_{jkl}(0)$ is non-zero when the first $k$ digits of bid $b_j$ (interpreted as a base $c$ number) is greater than or equal to the first $k-1$ digits of $b_0$ appended by $l$ (interpreted as a base $c$ number). These $v_{jkl}$ are used in round $k$ to determine the $k^{th}$ digit of the selling price by testing all possible values.

Since the $u_{jk}$ and $w_{jk}$ are computed from values $u_{j(k-1)}$, $v_{j(k-1)l}$, and $w_{j(k-1)}$ in round $k-1$, we must establish values for the $u_{j1}$ and $w_{j1}$. For all $j$, let $u_{j1}$ be the constant polynomial 0 and let $w_{j1}$ be the constant polynomial 1. In an implementation, computation using these initial values will be trivial (i.e. addition to 0 or multiplication by 1).

The auctioneers will perform summations of the $v_{jkl}$ over certain values of $j$ (i.e. subsets of $\{1, \ldots, n\}$). Define $\{P_y : 1 \leq y \leq \lceil \log n \rceil\}$ to be a collection of subsets of $\{1, \ldots, n\}$ (the index set for the bidders). Let $j \in P_y$ if and only if the $y^{th}$ bit (in a $\lceil \log n \rceil$ bit binary representation) of $j-1$ is zero. We will interpret these $P_y$ as partitions which separate $\{1, \ldots, n\}$ into two parts based on inclusion in the set $P_y$. The rationale behind the selection of these partitions (the $P_y$ for $1 \leq y \leq \lceil \log n \rceil$) is that for any pair of distinct bidders $j$ and $j'$, there is at least one partition which separates $j$ from $j'$.

### 4.2.1 Phase 1: Determining the $k^{th}$ digits of the bids

During the $k^{th}$ round of resolution, the auctioneers compute the $n(c-1)$ values

$$\forall j, l . v_{jkl} = u_{jk} + s_{jkl} \cdot w_{jk}.$$

The products $s_{jkl} \cdot w_{jk}$ require a degree reduction step on all $n(c-1)$ products (these can be done simultaneously). This step is necessary only once one of the first $k-1$ digits of the bid is determined to be non-zero. Prior to this point, the value of $w_{jk}(0)$ is known to be 1 for all $j$, and the multiplication is unnecessary (in particular, it will never be necessary in the first round).

### 4.2.2 Phase 2: Determining the $k^{th}$ digit of the selling price

Thew auctioneers now compute

$$\forall l . S_{kl} = \sum_{y=1}^{\lceil \log n \rceil} \left( \left( \sum_{j \in P_y} v_{jkl} \right) \left( \sum_{j' \notin P_y} v_{j'kl} \right) \right).$$

The inner summations may be performed without communication. Simple dynamic programming can be used to perform these summations in $3n(c-1)$ additions (of points in the field $Z_p$). The $(c-1)\lceil \log n \rceil$ multiplications require a degree reduction step on these $(c-1)\lceil \log n \rceil$ products. The outer summations may then be performed with $(c-1)(\lceil \log n \rceil - 1)$ field additions and no communication.

The outer summation runs over the indices for the partitions. The inner summations run over all $j$ in each half of a particular partition. These summations act like a logical OR operation over one side of the partition. In other words, if the partition side contains one or more bids which are at least as high as the test value, then the evaluation of this sum at 0 will be non-zero. For the polynomial result of multiplying the two inner summations, the evaluation at 0 will be non-zero only when the there is at least one bid on each side of the partition which is as high as the test bid. From this it is clear that no summand in the outer summation will have a value at 0 which is non-zero unless there are at least two bids which are at least as high as the test value. Conversely, we chose the partitions such that some partition would separate any pair $j, j'$, so if there are two bids which are greater than the test bid, then at least one summand has a value at 0 which is non-zero. This means that (with high probability, see section 4.8) $S_{kl}(0)$ will be non-zero if and only if there are at least two bids at least as high as the test bid, or, equivalently, that the selling price is at least as high as the test bid.

By this stage, the auctioneers must have agreed on two further shared polynomials for each value of $l$: one of degree $t$ (call it $R_{kl}$) which is selected uniformly randomly from $Z_p[x]$ and one of degree $2t$ (call it $R'_{kl}$) which is selected to be uniformly random up to the constraint that its value at 0 is 0. The $k$ subscript is used to indicate that different polynomials should be chosen for each round. Since they are dependent only on $m$, $c$, and $p$, the $R_{kl}$ and $R'_{kl}$ may be generated in batches and used over the course of many auctions.

The auctioneers compute $\forall l. S'_{kl} = R'_{kl} + S_{kl} \cdot R_{kl}$. This computation has no influence on the determination of the winner or selling price. Rather, it is necessary to preserve secrecy of the bids when $S_{kl}(0)$ is revealed. The need for this step is not entirely theoretical; if it is not performed, there is an attack by a combination of an auctioneer and some bidders which could potentially reveal certain information about competing bids. Revealing $S'_{kl}$ indicates only whether $S_{kl}(0)$ is zero or non-zero, which is exactly the information we will use to decide the $k^{th}$ digit of the selling price.

Now, the secrets of these $S'_{kl}$ are computed by the auctioneers (i.e., the values $\forall l. S'_{kl}(0)$). Let $l_0$ be the the largest value of $l$ for which $S'_{kl} \neq 0$ ($l_0 = 0$ if $S_{kl} = 0$ for all $l$). The $k^{th}$ digit of the bid is now set equal to $l_0$ (i.e. $b_{0k} \leftarrow l_0$).

#### 4.2.3 Phase 3: Updating control values $u$ and $w$

In preparation for the next round, the auctioneers assign

$$\forall j. u_{j(k+1)} = v_{jk(l_0+1)}, w_{j(k+1)} = v_{jkl_0}.$$

For the boundary conditions where $v_{jkl}$ would be undefined: if $l_0 = 0$ then $w_{j(k+1)} = w_{jk}$; if $l_0 = c-1$ then $u_{j(k+1)} = u_{jk}$. This requires no communication and trivial computation.

### 4.3 Determining the winning bidder

Consider the case that the highest bid is strictly higher than the second highest bid. In this case, for exactly one value of $j$, $u_{j(d+1)}(0)$ will be non-zero. Some agent (possibly the auctioneers) can collect the $u_{j(d+1)}$ polynomials and determine the winner. Since there can be only one bidder with a higher price, this reveals no extra information.

If there is a tie for the highest bid, then $\forall j. u_{j(d+1)}(0) = 0$, and for at least two values of $j$, $w_{j(d+1)} > 0$. These values of $j$ correspond to the tied bidders.

#### 4.3.1 Breaking ties

Handling the special case where several biddders tie for the highest bid is surprisingly complex. First, simply detecting that there has been a tie is an information leak. This means that perfect privacy requires a method of computing the winner of a tied or untied bid in a uniform manner. Since computation and communication costs would leak information if they were different for tied or non-tied cases, this means that the overhead costs for all auctions must be the same as the overhead for the worst-case tie-breaking situation.

The simplest approach is to reveal $\forall j. u_{j(d+1)}(0)$ as described above to determine if there is a clear winner. If all these values are 0, then there is a tie. Now reveal $\forall j. w_{j(d+1)}(0)$ and randomly select some $j$ for which $w_{j(d+1)} \neq 0$ as the winner. This method is direct and efficient, but reveals all tying bids (to the auctioneers).

Finding an efficient tie-breaking method to follow the bid resolution protocol as described is still an open problem. The authors are studying a modification to the overall protocol which allows efficient tie-breaking. The alternative has more complexity (in terms of its description) but uses a small value of $p$ (with other modifications to prevent errors), so that communication costs are reduced.

### 4.4 A sample auction

We will give a simple example at a medium level of detail to illustrate the basic operation of the computation. We will not describe the lower level details of secure distributed computation. Let $c = 2$ (the radix), and $d = 3$ (the number of digits). This will allow for 8 bidding points, say $\{\$10, \$20, \ldots, \$80\}$. With base $c = 2$, one polynomial is sufficient to represent each digit of the bid. This means that $l = 1$ for all polynomials, so we will omit the $l$ subscript. We will use $p = 7$, $m = 3$, $t = 1$, and $\forall i.\alpha_i = i$ for this example. Note that these parameters are chosen strictly for simplicity of the example; they are not secure.

Suppose three bidders $B_1$, $B_2$, and $B_3$ have bids $b_1 = \$20$, $b_2 = \$60$, and $b_3 = \$50$ respectively. In this case, the winner should be $B_2$ and the selling price should be the second highest bid, $b_3 = \$50$.

We use the symbols $\mathcal{T}$ and $\mathcal{F}$ to denote the values of polynomials' free coefficients. We will use the notation "$g = \mathcal{T}$" to mean that "$g(0) \neq 0 \pmod{p}$" and "$g = \mathcal{F}$" to mean that $g(0) = 0 \pmod{p}$.

During the bid submission step, each bidder selects the polynomials ($s_j$) to encode the digits of their bids. Each bid is encoded with one polynomial per digit (since $c = 2$). The bids are $b_1 \rightarrow s_{11}s_{12}s_{13} = \mathcal{F}\mathcal{T}\mathcal{F}$ and similarly $b_2 \rightarrow \mathcal{T}\mathcal{T}\mathcal{F}$ and $b_3 \rightarrow \mathcal{T}\mathcal{F}\mathcal{T}$.

The shared polynomials are distributed among the auctioneers, after which the auctioneers compute the result without further involvement from the bidders.

For example, the polynomials for $b_1$ could be $s_{11} = 4x + 0$, $s_{12} = 3x + 3$, and $s_{13} = 6x + 0$, and then bidder $B_1$ would share her polynomials by

- $B_1 \rightarrow A_1 \ : \ (4, 6, 6)$
- $B_1 \rightarrow A_2 \ : \ (1, 2, 5)$
- $B_1 \rightarrow A_3 \ : \ (5, 5, 4)$

Given the points of the shared polynomials, each auctioneer computes $v_{jk}$ at $k^{th}$ round by $v_{j(k+1)} = u_{jk} + s_{jk}w_{jk} \pmod{p}$. Table 1 shows how the polynomials are assigned through $k = 1, 2, 3$. The starting (i.e. $k = 1$) values are $\forall j.v_{j1} = s_{j1}, u_{j1} = \mathcal{F}, w_{j1} = \mathcal{T}$. As stated in the description, the values for the $u_{j1}$ are $w_{j1}$ just for consistency, they are known values and are only in the useless operations of multiplication by 1 ($\mathcal{T}$) or addition to 0 ($\mathcal{F}$).

- In round 1 ($k = 1$), there are at least 2 (exactly 2 in this case) $j$ for which $v_{1k} = \mathcal{T}$, and thus $S_1 = \mathcal{T}$. This determines that the first digit of the selling price is 1 ($b_0 = 1\_\_$ or equivalently $b_0 \in \{4, 5, 6, 7\}$). Following the updating rule provided in section 4.2.3, auctioneers compute new polynomials $w_{j2} = v_{j1}, u_{j2} = u_{j1}$.
- In round 2, there is only one $j$ for which $v_{2k} = \mathcal{T}$. So $S_2 = \mathcal{F}$, and the auctioneers update polynomials $\forall j.u_{j3} = v_{j2}, w_{j3} = w_{j2}$ accordingly. $S_2 = \mathcal{F}$ determines that the second digit of the selling price is 0, so $b_0 = 10\_2$ ($b_0 \in \{4, 5\}$).
- Finally, in round 3, there are two $j$ for which $v_{3k} = \mathcal{T}$ so the auctioneers determine that $S_3 = \mathcal{T}$. We have $S_1 = \mathcal{T}$, $S_2 = \mathcal{F}$, and $S_3 = \mathcal{T}$, so $b_0 = 101_2 = 5$ and the selling price is \$50. Since $S_3 = \mathcal{T}$, we have $\forall j.u_{j4} = v_{j3}$. For exactly one value $j_0$ we have $u_{4j_0} = \mathcal{T}$. The winning bidder is $B_{j_0}$. By combining their shares of $u_{jk}$ auctioneers determine that $j_0 = 2$; bidder $B_2$ is the winner.

### 4.5 Efficiency

In brief, the costs are non-trivial, and this protocol would not be appropriate for very low value (measured in cents) or extremely time-sensitive (results needed in seconds) auctions. However, most real-world auctions do not fall into these categories. Although the costs are not competitive with auctions which do not preserve privacy, such as [7], we believe that the complete privacy of the bids (and, if desired, the bidders)—even after the auction is completed—could well justify the higher implementation costs in many situations.

A major factor in the cost of the protocol is the fault tolerance. We describe the costs for $3t < m$, but if we were to accept slightly lower fault tolerance, the low-level implementation would be greatly simplified, resulting in a substantial savings in communication costs. For example, changing $m$ from 4 to 5 (and making the improvements that this would allow) would reduce the volume of communication per auctioneer by nearly a factor of 10 in the examples in section 4.5.2.

Communication costs will be the dominant concern for the protocol described, and will consist largely of messages containing many points drawn from the field $Z_p$. To simplify the analysis, we will consider the costs in the case where there are no attempts to cheat; the cost to detect and correct cheating depends heavily on the number of attempted cheaters. Any cheating attempt (by at most $t$ auctioneers) will be immediately detected and corrected, so cheating attempts by the auctioneers are unlikely to be frequent. Additionally, we will not

| $k=1$ | $b_j$ | $s_j$ | | | $u_j$ | | | $v_j$ | | | $w_j$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $B_1$ | 2 | $\mathcal{F}$ | $\mathcal{T}$ | $\mathcal{F}$ | $(\mathcal{F})$ | | | $\mathcal{F}$ | | | $(\mathcal{T})$ | | |
| $B_2$ | 6 | $\mathcal{T}$ | $\mathcal{T}$ | $\mathcal{F}$ | $(\mathcal{F})$ | | | $\mathcal{T}$ | | | $(\mathcal{T})$ | | |
| $B_3$ | 5 | $\mathcal{T}$ | $\mathcal{F}$ | $\mathcal{T}$ | $(\mathcal{F})$ | | | $\mathcal{T}$ | | | $(\mathcal{T})$ | | |

$S_1 = \mathcal{T}$

| $k=2$ | $b_j$ | $s_j$ | | | $u_j$ | | | $v_j$ | | | $w_j$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $B_1$ | 2 | $\mathcal{F}$ | $\mathcal{T}$ | $\mathcal{F}$ | $(\mathcal{F})$ | $(\mathcal{F})$ | | $\mathcal{F}$ | $\mathcal{F}$ | | $(\mathcal{T})$ | $\mathcal{F}$ | |
| $B_2$ | 6 | $\mathcal{T}$ | $\mathcal{T}$ | $\mathcal{F}$ | $(\mathcal{F})$ | $(\mathcal{F})$ | | $\mathcal{T}$ | $\mathcal{T}$ | | $(\mathcal{T})$ | $\mathcal{T}$ | |
| $B_3$ | 5 | $\mathcal{T}$ | $\mathcal{F}$ | $\mathcal{T}$ | $(\mathcal{F})$ | $(\mathcal{F})$ | | $\mathcal{T}$ | $\mathcal{F}$ | | $(\mathcal{T})$ | $\mathcal{T}$ | |

$S_2 = \mathcal{F}$

| $k=3$ | $b_j$ | $s_j$ | | | $u_j$ | | | $v_j$ | | | $w_j$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $B_1$ | 2 | $\mathcal{F}$ | $\mathcal{T}$ | $\mathcal{F}$ | $(\mathcal{F})$ | $(\mathcal{F})$ | $\mathcal{F}$ | $\mathcal{F}$ | $\mathcal{F}$ | $\mathcal{F}$ | $(\mathcal{T})$ | $\mathcal{F}$ | $\mathcal{F}$ |
| $B_2$ | 6 | $\mathcal{T}$ | $\mathcal{T}$ | $\mathcal{F}$ | $(\mathcal{F})$ | $(\mathcal{F})$ | $\mathcal{T}$ | $\mathcal{T}$ | $\mathcal{T}$ | $\mathcal{T}$ | $(\mathcal{T})$ | $\mathcal{T}$ | $\mathcal{T}$ |
| $B_3$ | 5 | $\mathcal{T}$ | $\mathcal{F}$ | $\mathcal{T}$ | $(\mathcal{F})$ | $(\mathcal{F})$ | $\mathcal{F}$ | $\mathcal{T}$ | $\mathcal{F}$ | $\mathcal{T}$ | $(\mathcal{T})$ | $\mathcal{T}$ | $\mathcal{T}$ |

$S_3 = \mathcal{T}$

Table 1: Computation of polynomials for example auction

consider some small messages (mostly involving optional counter-signatures on the bid submissions) which do not contribute substantially to the total bandwidth used.

The following efficiency analysis assumes that the protocol uses the absolute secret verification as described in [2, 1]. Each polynomial is distributed by sending two degree-$t$ polynomials to each recipient. Each recipient then shares 1 point with each other recipient to verify the secret. As with much of the communication in the protocol, many points can be verified in parallel.

The multiplication and degree-reduction of a pair of polynomials can be compressesd into two rounds of communication (in addition to some preparation which may occur prior to the protocol and which is described below). In the first phase, each auctioneer verifiably distributes shares for a total of $t+3$ polynomials: polynomials encoding his shares of the two polynomials to be multiplied, a polynomial encoding his share of the result of the multiplication, and $t$ other polynomials which enable confirmation that the product was correctly computed on that share. In the second phase, all the check points are exchanged among the auctioneers, ensuring that the secrets from the first phase are valid. Also, a different set of check values is broadcast by each auctioneer ($4t$ points each). These points ensure that a consistent (across all auctioneers) set of shares was used to compute the product polynomial. Given the honest majority, this ensures correct computation and enables correction of any errors (i.e. attempts to cheat). These two rounds combined require the communication of $2t^2+(m+7)t+3m+3$ for each ordered pair of auctioneers, and an additional $4t$ points of broadcast per auctioneer. The product of many pairs of polynomials may be computed in parallel.

The preparation for multiplication, as referred to above, is that the auctioneers must generate a random degree-$t$ shared polynomial to be used for each multiplication. These polynomials can be generated in large batches prior to a run (or many runs) of the protocol. The process is simply for each auctioneer to generate and verifiably share one such polynomial, the sum of all $m$ shared polynomials is one suitable random polynomial (even if $t$ auctioneers are corrupt).

### 4.5.1 Costs for protocol phases

Given this verification model above, a bid submission consists of $m$ messages (one to each auctioneer) from the bidder, each of size $2(t+1)d(c-1)$ points. The auctioneers then verify the submission by an all-to-all communication with messages of size $d(c-1)$ points.

Determining a digit of the selling price causes two occasions for degree reduction: the first when computing the $v_{jkl}$ in phase 1, the second when computing the summands of $S_{kl}$ in phase 2. This is a total of $(c-1)\cdot(n+\lceil\log n\rceil)$ multiplications over 4 communication rounds. There is one additional communication round where each auctioneer broadcasts $c-1$ points to reveal the $S'_{kl}$ shares.

### 4.5.2 Example costs

Let us assume we have $m = 4$ auctioneers (so $t = 1$, no single auctioneer can cheat or violate privacy), and let $\log p \approx 96$, so each point is 12 bytes. We will estimate the communication costs (in terms of rounds, messages, and bytes) for two sample auctions. We will assume that all bids must be verified immediately (this maximizes the number of messages needed during bid submission), that the leading digit of the selling price is non-zero (worst-case). Costs will be counted as the total of sent and received for both bytes and messages.

Example 1: Consider the case with $n = 1000$ bidders, $V = 1024$ possible prices, and base $c = 2$ bid encoding (so $d = 10$).

- Bid submission. Each bidder sends 4 messages of 480 bytes each. Each auctioneer receives one of these messages and then sends 120 bytes to each other auctioneer. Total communication for each auctioneer during submission is 1.2MB in 7000 total messages.

- Bid resolution. In the first round of the protocol, phase 1 is unnecessary, so there will be 10 multiplications over 3 communication rounds. The remaining 9 protocol rounds each require 1010 multiplications and 5 communication rounds. Each multiplication requires 480 bytes of communication between each pair of auctioneers. Total communication for each auctioneer is $9100 * 960 * 3 = 26.2$MB in 288 messages over 48 communication rounds.

Example 2: Consider the case with $n = 50$ bidders, $V = 1024$ possible prices, and base $c = 4$ bid encoding (so $d = 5$).

- Bid submission. Each bidder sends 4 messages of 720 bytes each. Each auctioneer receives one of these messages and then sends 180 bytes to each other auctioneer. Total communication for each auctioneer during submission is $90K$ bytes in 350 total messages.

- Bid resolution. In the first round of the protocol, phase 1 is unnecessary, so there will be 18 multiplications over 3 communication rounds. The remaining 4 protocol rounds each require 168 multiplications and 5 communication rounds. Each multiplication requires 480 bytes of communication between each pair of auctioneers. Total communication for each auctioneer is $690 * 960 * 3 = 2.0$MB in 138 messages over 23 communication rounds.

### 4.5.3 Asymptotic efficiency

This protocol we describe uses $O(t^2 n \log V)$ communication for each auctioneer. Information theory tells us that the bids must be $O(\log V)$ in size. So, if we guarantee that $t$ or fewer auctioneers can obtain no information about the bids, each bidder must distribute $O(t \log V)$ information to the auctioneers (which is $O(n \log V)$ per auctioneer). The verifiable secret sharing methods actually require an extra factor of $O(t)$ above the theoretical minimums for non-verifiable secret sharing. So, in total, just distributing the bids requires $O(tn \log V)$ communication for each auctioneer.

There are two major constant factors hidden by the notation which lie at the heart of high communication costs. The first is the large field we have chosen; the second is the constant associated with multiplying two polynomials. The choice of a large field was made to reduce the number of rounds of communicaiton required in phase 2, and alternative are being explored. The constant associated with multiplication is a more fundamental result of the distributed computation techniques used and the high fault-tolerance ($3t < m$) allowed.

## 4.6 Accountability

Given the application of these protocols to electronic commerce, we wish to ensure that the participants can be held accountable for their actions. By this, we mean that there are certain economic or legal consequences to the actions of the participants, and that there is sufficient non-repudiable evidence to prove what actions were taken and to enforce these consequences.

### 4.6.1 Deposits

In order to support non-repudiation, or at least to associate a cost with repudiation, bid submissions may include some electronic payment (e.g., digital cash) as a form of deposit. This is particularly useful in cases where there would be a difficulty holding the bidders accountable through a legal process (e.g., if the bidders are completely anonymous). The amount of the deposit may or may not depend on the value of the bid. If a bidder fails to accept the good, the deposit is forfeited and the auction is repeated. Depending on the particular payment system used for the deposit, some safe-guarding techniques may be used to prevent cheaters from redeeming deposits inappropriately. For example, verifiable signature sharing [6] may be used to require some threshold number of auctioneers to agree to redeem a digital cash deposit. The use of verifiable signature sharing and digital payment deposits on bids first appears in [7].

There is a cost associated with such deposits beyond the computational cost of the implementation. The deposit is "in use", and is unavailable for other purposes during the course of the auction. The precise effect of having some amount in use is dependent on the digital payment scheme being used. In an anonymous digital cash scheme, for example, it is likely to mean that the funds have been withdrawn from a (possibly interest bearing) account. In any scheme, there is an opportunity cost to having the funds in use which must be taken into account. If the overall opportunity cost to the bidders is high (e.g. if the deposit is large or if there are many bidders), this will tend to discourage bidding and depress the selling price. Therefore, compensation for the auctioneer and cost to the bidders must be balanced when choosing the value of the deposit.

One implementation on deposits is for the seller to set fixed value for all bidders as the penalty for default. The appropriate magnitude for this default penalty is highly dependent on the specific characteristics of the seller, the good, and the bidders. This method is optimal in terms of the computation and communication costs and may be the best in practice.

Another method, not detailed here, would be to have the deposit match the value of the bid. This would introduce further computational complexity, particularly since the auction is second-price and so determining the value of the deposit, even for the winning bidder, would violate privacy. An efficient method for such deposits is an open problem (see section 5) Even discounting the computational overhead, this method yields optimal results only if the total of opportunity costs of all deposits is trivial.

### 4.6.2 Bidder accountability

The signature on the bid submission enables the auctioneers to prove the validity of a bid even if the majority of the auctioneers are not trusted. In the non-anonymous case, a well-known public key for the bidder should be used. In the anonymous case, the asymmetric key should be registered with some escrow agency or cryptographically linked to a deposit. Whatever key is used, the meaning of "bidder" in the following discussion is simply the entity which can generate signatures with the correct key. In all cases, the auctioneers must be able to prove that the bidder (by means of her asymmetric key) placed a bid at least as high as the selling price.

The simplest means of resolving a dispute is for the auctioneers to send all components to a trusted authority. The accuracy of the components can be verified by the signatures, and then the precise bid can be determined. The drawback of this method of resolution is that the authority will learn the exact value of the bid. However, for the auctioneers to inaccurately determine a winner, there must be a coalition of auctioneers large enough to determine the bid themselves, each member of which is corrupt or faulty. So the bid may already be known to the corrupt auctioneers.

Another method for resolving such a dispute is to perform a greatly simplified version of the auction. This simplified auction may be performed as a check by the auctioneers, to ensure that the winner was not selected incorrectly. Additionally, the shares possessed by the auctioneers (signed by the bidder) may be transferred to some other distributed authority, which will perform the check to resolve disputes. This simplified auction is performed with 3 bids: the disputed "winning" bid and two known bids $b_0$ and $b_0 - 1$ (where $b_0$ is the selling price determined by the disputed auction). If this auction returns a selling price of $b_0$, then the auction was correct; if it returns a selling price of $b_0 - 1$, it was incorrect. The result of the auction is always $b_0$ or $b_0 - 1$ and thus reveals no information other than whether the questioned bid is less than $b_0$.

### 4.6.3 Auctioneer accountability

The bid submissions can be signed (i.e. countersigned) by the receiving auctioneers and the signature made available to the submitting bidder. Then, if the selling price is published, any bidder who bid higher than the selling price can prove his superior bid. The same techniques described in section 4.6.2 for bidder accountability could be used to check that the bid is higher than the selling price without directly revealing the bid. In this case the two known bids for the simplified auction should be $b_0 + 1$ and $b_0$.

In this manner a bidder can prove a superior bid even if all auctioneers were corrupted (assuming that the bid was accepted in the first place). This method does not by itself protect against a denial-of-service attack by a large coalition ($> t$) of corrupt auctioneers unless they can be forced to accept and acknowledge a bid by some third party.

### 4.6.4 Anonymity

The protocol is designed to determine a selling price without revealing any of the bid values (other than the second highest, of course). The identities of the bidding parties can also be protected. Anonymous (or pseudonymous) public keys may be used if there is some form of accountability to the keys which is considered acceptable for the purposes of the auction (possibly just a deposit as in 4.6.1). To preserve anonymity over real networks, the communications channel between the bidders and the auctioneers must be anonymized by means of some anonymous forwarding system.

### 4.7 Passive attacks

Due to the use of verifiable secret sharing, no coalition of at most $t$ auctioneers can determine any information about the bidding from their shares of the bids. Similarly, the degree reduction steps preserve secrecy against coalitions of at most $t$ auctioneers. But what about the polynomials $S'_{kl}$ which are revealed in the course of determining the selling price? A single $S'_{kl}$ is uniformly random and independent of all other variables except for its free coefficient. If $S_{kl}(0) = 0$, then $S'_{kl} = 0$. If $S_{kl} > 0$, then $S'_{kl}(0)$ is an element uniformly distributed over $Z_p \setminus \{0\}$. Note that $S_{kl}(0) > 0$ exactly when there are at least two bids whose value is at least the speculative selling price (the previously determined digits of $b_0$ together with $k^{th}$ digit $l$).

### 4.8 Error analysis

During the course of computation, we occasionally assume that the sum of several uniformly random and independent non-zero values in $Z_p$ is also non-zero. The probability that such a sum is 0 is at most $\frac{1}{p-1}$. Such errors may occur in the computation of the $v_{jkl}$ or $S_{kl}$.

In the case of computing $v_{jkl}$, such a chance will introduce an error into the result of the bidding only if it occurs for the $l$ that are relevant to the selection of $b_0$ (i.e., $l_0$ or $l_0 + 1$)). Also, $u_{jk}(0)$ will never be non-zero for any bidder other than the winner, so this error will be possible only for the winning bidder. The non-zero $s_{jkl}$ are uniformly random (from $Z_p \setminus \{0\}$). They are also independent of $s_{jk'l}$ for $k' \neq k$, and thus are independent of the $u_{jk}$ and $w_{jk}$ (which are functions of the $s_{jk'l}$ with $k' < k$). Therefore, the value $s_{jkl} \cdot w_{jk}$ is uniformly random (from $Z_p \setminus \{0\}$). So the sum of $u_{jk}$ and $s_{jkl} \cdot w_{jkl}$ (for any particular $k$ and $l$) can equal zero with probability at most $\frac{1}{p-1}$.

Since the non-zero $s_{jkl}(0)$ are uniform and independent, the $v_{jkl}(0)$, if non-zero, will also be uniform and independent of any $v_{j'kl}(0)$ which is non-zero (for $j' \neq j$). In other words, the only dependency is contained in the property of being non-zero.

In the case of the $S_{kl}$, an error in the result of the bidding will be caused only for the one relevant $l$ value $l_0$. The chance of incorrectly generating a $S_{kl}$ with value 0 at 0 is at most $\frac{2}{p-1}$. To see this consider the values of $S_{kl}$ as we compute it over increasing subsets of the bidders. Note the change in $S_{kl}(0)$ when we include the final bidder whose bid is as high as the test value (i.e. for whom $v_{jkl}(0) \neq 0$).

We can bound these sources of error by $\frac{2d}{p-1}$ from the $v_{jkl}$ and $\frac{2d}{p-1}$ for the $S_{jk}$, totalling $\frac{6d}{p-1}$. We must choose $p$ large enough that $\frac{4d}{p-1}$ is negligible. Since $d = \log_c V$, where $V$ is the number of bidding points, its value in any implementation would be small (certainly less than 40). Making a tradeoff between speed and possibility of error, $p$ should be in the range of 64-128 bits.

## 5 Open Problems

There are a variety of natural questions left unanswered by the work described. Below, we list a few natural directions for further work in this area. Work is under way on variations on the protocol which will address tie-breaking issues, reduce communication costs, and

1. *Tie-breaking*
   The protocol described does not provide efficient tie-breaking without some loss of privacy.

2. *Communication Costs*
   While we believe the communication costs are sufficiently small to make this protocol practical in many situations, low-value auctions are likely to play an increasingly important role in electronic commerce. Efficiency improvements that enable auctions with private bids in these low-value situations could be very useful.

3. *Hierarchical Auctions*
   In some situations, it might be advantageous to hold sub-auctions which partially determine the outcome of the auction. For example, each country might hold a sub-auction, with the leading candidates from each country participating in a final auction. In order to preserve privacy, the winners of the sub-auctions and their bids must not be revealed.

4. *Double Auctions and Auction Markets*
   A double auction is a more general form of auction where there are multiple sellers and multiple buyers. All parties tender bids and a market

clearing price is determined from those bids. A market clearing price is the equilibrium price at which the supply and demand (in units of the good) are equal. An auction market is a generalization of a double auction to continuous time. New bids are added and removed over time, causing the market clearing price to fluctuate. The stock market is a well known example of an auction market. Double auctions and auction markets are powerful market mechanisms, and privacy protecting protocols for these mechanism would be desirable. However, to be useful, such protocols must be highly efficient, particularly in the case of auction markets.

5. *Privacy vs. Performance*
   The protocol described protects information privacy at the cost of greater overhead. It is possible that performance could be improved by relaxing some of the privacy constraints. What is the nature of this tradeoff? Depending on the context of the auction, relaxing privacy constaints could be appropriate.

# References

[1] D. Beaver. Security, fault tolerance, and communication complexity in distributed systems. Technical Report HAR-CS-24-90, Computer Science Department, Harvard University, Boston, MA, 1992. Ph.D. thesis.

[2] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computing. In *Proceedings of the* $20^{th}$ *STOC*, pages 1–10. ACM, 1988.

[3] D. Chaum, C. Crepeau, and I. Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the* $20^{th}$ *STOC*, pages 11–19. ACM, 1988.

[4] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *STOC*, pages 383–395. ACM, 1985.

[5] David Clark. Internet cost allocation and pricing. In Lee McKnight and Joseph P. Bailey, editors, *Internet Economics*, Cambridge, MA, 1997. MIT Press.

[6] Matthew K. Franklin and Michael K. Reiter. Verifiable signature sharing. In L.C. Guillou and J. Quisquater, editors, *EUROCRYPT95*, pages 50–63. Springer-Verlag, 1995. Lecture Notes in Computer Science No. 921.

[7] Matthew K. Franklin and Michael K. Reiter. The design and implementation of a secure auction server. *IEEE Trans. on Software Engineering*, 22(5):302–312, 1996.

[8] O. Goldreich, S. Micali, and A. Wigderson. How to play and mental game. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229, May 1987.

[9] Andreu Mas-Colell, Michael D. Winston, and Jerry R. Green. Incentives and mechanism design. In *Microeconomic Theory*, New York, 1995. Oxford University Press.

[10] Lee McKnight and Joseph Bailey. *Internet Economics.* MIT Press, Cambridge, MA, 1997.

[11] Paul Milgrom. Auction theory. In *Advances in Economic Theory 1985: Fifth World Congress.* Cambridge University Press, 1985.

[12] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–614, November 1979.

[13] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16, 1961.

[14] Robert Wilson. Auction theory. In J. Eatwell, M. Milgate, and P. Newman, editors, *The New Palgrave*, London, 1987. MacMillan.

# A Resilient Access Control Scheme for Secure Electronic Transactions

Jong-Hyeon Lee [*]
*Computer Laboratory, University of Cambridge*
*Pembroke Street, Cambridge CB2 3QG, England*
Jong-Hyeon.Lee@cl.cam.ac.uk
http://www.cl.cam.ac.uk/~jhl21

## Abstract

There have been many studies of the management of personal secrets such as PIN codes, passwords, etc., in access control mechanisms. The leakage of personal secrets is one of the most significant problems in access control. To reduce such risks, we suggest a way of authenticating customers without transferring explicit customer secrets. Furthermore, we give a secure online transaction scheme based on our access control mechanism.

Needham gave an example of Personal Identification Number (PIN) management for banking systems [Nee97] that presented a way to control PIN codes. It inspired us to develop an access control model for electronic transactions which enforces a strict role definition for personal secret generation and maintenance. We extend it to a payment model. Our scheme provides enhanced privacy for customers, non-repudiation of origin for the customer order and payment transactions, and protection from personal secret leakage. Since it does not rely on either public key cryptosystems or auxiliary hardware such as chip cards and readers, its deployment within existing environments could be cost-effective.

## 1 Introduction

For user authentication, traditional passwords or PIN codes are still one of the most common methods, although they are notoriously vulnerable to attacks. Even systems with sophisticated security protocols often employ such authentication procedures, since they are easy and familiar to users. For this reason, many studies of personal secret management have been done, like strengthening passwords [ALN97], enhancement of existing password protocols [BM92, BM93], protection for poorly chosen passwords [GLNS93], one-time password schemes [Hal94, Rub96], and so on.

In access control, we have to consider who generates secrets and who maintains them. Some banks generate customers' PIN codes, and send them to customers. Some on-line shops ask customers to generate passwords for their services, and keep the passwords in their database. It is common that personal secrets are known to service providers such as banks or on-line shops, as shared secrets between the service providers and customers. But there is no reason for customers to trust service providers; this is dangerous.

Needham gave a simple PIN code management scheme for banking systems [Nee97], which shows a strict role definition of PIN generation and management. The PIN codes are generated and maintained by customers themselves. This idea can be a basis for building a privacy-enhanced transaction model with strict role definition.

One of the most frequently asked question in se-

[*] This work is supported in part by the EPSRC, under grant number GR/L95809 on Resilient Security Mechanisms. The views and conclusions in this paper are that of the author alone.

curity is protection versus cost. It is a good idea to have alternatives with the same function at different levels of cost. Public key cryptosystems provide high confidentially, authentication, etc., but they require a public key infrastructure. For some applications, it is not a proper solution to adopt a public key cryptosystem because of the cost.

There have been many attempts to use hash functions rather than public key cryptosystems. IBM's KryptoKnight is such an example [MTvHZ92]. From a user's perspective, KryptoKnight provides services and facilities which are very similar to those of Kerberos [NT94] and based on the well-known Needham-Schroeder scheme [NS78], but it uses hash functions and Message Authentication Codes (MACs). We also use only one-way hash functions to build an access control mechanism and a payment mechanism. Since it is based on hash calculations and nonces, the required infrastructure is lighter than that of a public key cryptosystem, to which it can provide a low cost alternative.

We will develop a customer-oriented transaction model in which personal secrets are generated and maintained by customers. It supports customer registration both in a bank and in an online shop, and a transaction procedure between three principals; a customer, a merchant, and a bank.

## 2 Analysis of Needham's Example

Needham's PIN code management system differs from current ATM systems in the generation and handling of the PIN code. Most ATM systems use encryption to protect customers' PINs. The details vary from one bank to another, but many use variants of a system originally developed by IBM, in which the PIN is derived from the account number by encryption. That is, PINs are generated by banks. This makes it difficult to resolve disputes between banks and their customers [And94].

In Needham's proposal, banks do not know customers' PINs. A PIN code is generated by a customer herself, and is not stored by the bank. From the bank's point of view, this feature liberates it from the responsibility for internal leakage of PINs and assurance of their fair and truly random generation. Banks thus have a defence against an allegation that they negligently permitted the PIN to become known. For customers, they can obtain more privacy by generating and keeping their own PINs for themselves.

Needham's scheme is described under the assumption that a customer has a personal computer and a card writer. Let $H$ be a one-way hash function. To initialise the scheme, the customer writes on the card a random number $r$ and a hash $H(n, b)$ of her name $n$ and date of birth $b$. She writes $H(n, b)$ and $H(r, pin)$ on a floppy disk, where the $pin$ is chosen by herself. She then takes the floppy to the bank and says "Please connect $H(r, pin)$ to my personal details $H(n, b)$ and my account number 401608 80614874".

When a cash machine accepts her card, it reads $H(n, b)$ and $r$, works out $H(r, pin)$ where $pin$ is the PIN as entered, and sends the value $H(n, b)$ and $H(r, pin)$ to the bank. Note that the PIN is never sent to the bank and as $r$ is random it cannot be recovered from $H(r, pin)$. The bank looks up $H(r, pin)$ and if it is found, the table yields $H(n, b)$ for checking as well as the account number.

In this example, there are two principals and one intermediate: a customer and a bank are principals and an ATM is an intermediate. A banking transaction is performed between a customer and a bank, and authentication procedures are done by an ATM and a bank. The ATM checks the validity of a card holder by PIN entered, the bank checks a customer's information and her account number by $H(r, pin)$.

The separation of capability between principals is well defined, but there is no such separation between the customer and the ATM. The assumption that the PIN is never sent to the bank even in encrypted form requires the customer trusts the ATM and its operator. The role of the ATM is more important than at present and a false-terminal attack using a corrupted ATM is still possible. However this is inherent in the use of magnetic strip technology.

A replay attack for the pair of hash values $(H(n, b), H(r, pin))$ is possible on a communication line between an ATM and a bank. If an

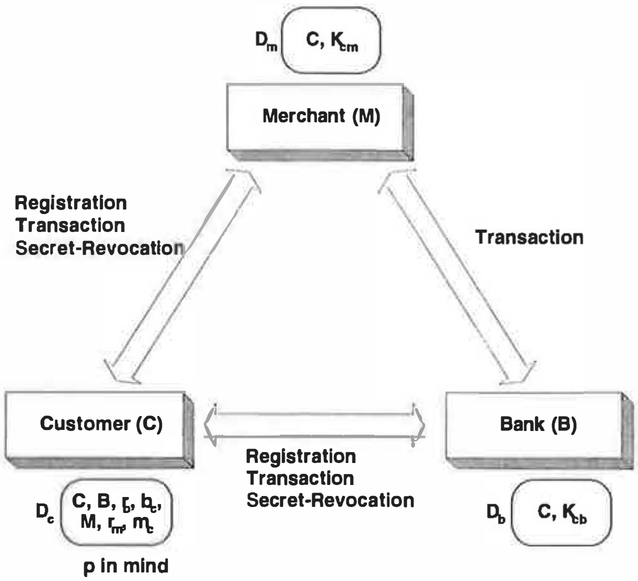


Figure 1: Principals and their interfaces

attacker takes the hash pair from the line, he can replay this pair on the same line and be authorised as a valid customer by a bank. Since the two values are always the same, they can be used at any time by attackers. We could prevent such attacks, of course, by installing a time-varying encryption or authentication mechanism on the line between the bank and ATMs. However, the bank would then join the ATM inside the customer's trust perimeter, and the value of the scheme would be lost. Against replay attacks, we use a nonce for each transaction in our payment scheme.

# 3 The Resilient On-line Transaction Scheme

We construct an on-line transaction scheme based on the analysis in the previous section. We define three procedures for access control and electronic transactions: a registration procedure, a transaction procedure, and a secret-revocation procedure. The principals in these transactions are a customer $C$, a merchant $M$, and a bank $B$. Merchants include Internet shopping malls, book sellers, on-line travel agents, news providers, etc.

We describe four procedures with brief protocol flows. In the protocol description, the notation $A \Longleftrightarrow B$ stands for protocols between $A$ and $B$, $A \rightarrow B$ a message from $A$ to $B$. The symbol $:=$ denotes an assignment. The notation $(x, y)$ is the concatenation of $x$ and $y$. Figure 1 shows the interfaces between the principals.

## 3.1 The registration procedure

Let us assume that a customer has an account with a bank. The registration procedure has two steps: a registration with a bank and a registration with a merchant. First, the customer generates a random number $r_b$, chooses a secret $p$, and calculates a hash $C := H(n, b)$ of her name $n$ and date of birth $b$. If the combination of $n$ and $b$ is not sufficient to identify a customer from others, one may add more detailed information. The customer writes $r_b$, $B$, and $C$ on a diskette for private use, say $D_c$, and writes $C$ and $K_{cb} := H(r_b, p)$ on a different diskette for registration, say $D_b$. Then she sends $D_b$ to the bank $B$ with her account number $a$, e.g. by registered mail or personal delivery to a branch of the bank.

When the diskette $D_b$ is received, the bank makes a link between the customer's account and $K_{cb}$, and sends an acknowledgement with a random $b_c$ back to the customer. The customer stores $b_c$ on her private diskette $D_c$. This is the registration procedure with the bank.

The registration procedure with the merchant is as follows: The customer generates a random number $r_m$, stores it on her private diskette $D_c$, and then sends the merchant a diskette $D_m$ containing $C$ and $K_{cm} := H(r_m, b_c, p)$ in the same way as in the above procedure. Then the merchant registers the customer's information on his database, and sends an acknowledgement with a unique merchant secret $m_c$ back to the customer. $m_c$ is a uniquely issued value for each customer, and will be used for verification of the merchant in transactions by a bank. The customer stores $m_c$ on $D_c$, calculates $K_{mb} := H(m_c, K_{cb})$, and sends $(C, m_c)$ with the merchant name $M$ to the bank. Then the bank constructs $K_{mb}$ with $m_c$ and $K_{cb}$, and adds $K_{mb}$ and $M$ to the customer's information.

In this procedure, since we have not assumed secure communication paths between the customer and the merchant/bank, we used physical transfer of shared secrets by diskette. If a secure path is available such as SSL/TLS [DA97] or SSH [Ylö96], we can replace diskette transfer by such a path. As a further alternative, the customer can send $K_{cm}$ to the merchant in a physical transaction between her smartcard and the merchant terminal.

Thus the customer can establish a relationship with a merchant either when she is on the merchant's premises, or when she has a secure link to the merchant, or when the bank is on-line. At the same time, the customer could establish a payment limit for the merchant (though we omit the details).

In some cases like closed user group services, the merchant needs to authenticate the customer's eligibility for the service. During the registration procedure, the merchant can request appropriate information such as a membership, age, etc., for the verification and provide classified services in the transaction procedure up to customers' eligibility.

**Protocol**

$\boxed{C \Longleftrightarrow B}$

$C$: generates $r_b$ and $p$
calculates $C := H(n, b)$ and $K_{cb} := H(r_b, p)$
stores $C$, $B$ and $r_b$ on diskette $D_c$
stores $C$ and $K_{cb}$ on diskette $D_b$

$C \to B$: $D_b, a$

$B$: stores $C$ and $K_{cb}$ w.r.t. $a$
generates a random number $b_c$

$B \to C$: $b_c$

$C$: stores $b_c$ on $D_c$

$\boxed{C \Longleftrightarrow M}$

$C$: generates $r_m$
stores $M$ and $r_m$ on diskette $D_c$
calculates $K_{cm} := H(r_m, b_c, p)$
stores $C$ and $K_{cm}$ on diskette $D_m$

$C \to M$: $D_m$

$M$: generates account for $C$ with $K_{cm}$
generates a random number $m_c$

$M \to C$: $m_c$

$C$: stores $m_c$ on $D_c$
calculates $K_{mb} := H(K_{cb}, m_c)$

$C \to B$: $C, M, m_c$

$B$: constructs $K_{mb}$ with $m_c$ and $K_{cb}$
adds $K_{mb}$ and $M$ to $C$'s information

## 3.2 The client software distribution

It is necessary for the customer to trust the client software she uses. This client software can be a plug-in for web browsers, a in-house program, etc., and obtained by download, mail delivery, or whatsoever. For such software, the specification should be public, and anyone can provide it. In theory, the customer could implement the software for herself, so she can trust her code and does not need to consider malicious action or compromise during download. In practice, she has to obtain the software from vendors she can trust. Whoever the code provider is, it should be guaranteed that the code does not contain malicious and erroneous code. The way to achieve proper verification is a controversial and interesting topic, and so is trusted distribution. Neither is a main focus of our paper.

## 3.3 The transaction procedure

The transaction procedure is as follows: When a customer wants to make a transaction, she establishes a connection to the merchant by invoking the client software. This software asks the customer to type the secret $p$, and establishes a *connection*[1] to the merchant. Then the merchant generates a transaction identifier $t$ and sends it back to the customer. This value is a nonce for preventing from replay attacks, and it consists of transaction time, date, and a serial number. After reading the random number $r_m$ from the diskette $D_c$, the software calculates $H(t, K_{cm})$, and sends $(C, H(t, K_{cm}))$ to the merchant. It should be guaranteed that the client program does not send the customer secret $p$ to the merchant.

The merchant looks up his database with searching key $C$, and checks validity of $H(t, K_{cm})$ for customer authentication. If it is valid, the merchant calculates $H(t, m_c, K_{cm})$, and sends it to the customer with an acknowledgement. After accepting the acknowledgement, the customer checks the hash value with $m_c$ for merchant authentication.

When the customer makes an order, the merchant requests her to pay, and sends a payment serial number $y$ and payment amount $u$. The customer generates a nonce $n_c$, calculates $T_{cmb} := H(t, y, u, K_{mb}, K_{cb})$, and confirms payment to the merchant by sending $(C, B, a, u, y, n_c, T_{cmb})$. The merchant asks the bank $B$ to clear the amount, by sending $(M, C, a, u, t, y, n_c, T_{cmb})$. If $T_{cmb}$ is correct, the bank transfers the money to the merchant with an acknowledgement [2] including $H(n_c, b_c, K_{cb})$. The value $H(n_c, b_c, K_{cb})$ is used by the merchant to generate an acknowledgement ticket for the customer. The bank has to store $T_{cmb}$'s that it has received for a pre-defined period in order to detect replays. In current credit card systems, the merchant must request that the bank clear transactions within a specified period from the transaction date - such as 21 days in UEPS/VISA COPAC [And92] - otherwise payment will not be made. The storage of $T_{cmb}$ does not require much resource. After receiving the acknowledgement from the bank, the merchant calculates $T_{bmc} := H(t, u, K_{cm}, H(n_c, b_c, K_{cb}))$ and sends this value as an acknowledgement to the customer. The customer checks this value, which can be used as a proof of the purchase order and of a proof of payment in case of dispute.

We may consider a false merchant who wants to cause problems for the real merchant by taking orders and not delivering. Since $T_{cmb}$ contains $K_{mb}$, he cannot gain monetarily. Furthermore, he has to send $T_{cmb}$ containing $K_{cm}$ back to the customer, so that the customer can check the eligibility of the merchant. Such a false merchant cannot step into the transaction.

**Protocol**

| | |
|---|---|
| $C \to M$: | service invocation via client software |
| $M$: | generates a transaction id $t$ |
| $M \to C$: | $t$ |
| $C$: | calculates $H(t, K_{cm})$ |
| $C \to M$: | $C, H(t, K_{cm})$ |
| $M$: | checks $H(t, K_{cm})$ |
| | calculates $H(t, m_c, K_{cm})$ |
| $M \to C$: | $H(t, m_c, K_{cm})$ |
| $C \to M$: | makes an order |
| $M \to C$: | $y, u$ |
| $C$: | generates a nonce $n_c$ |
| | calculates $T_{cmb} := H(t, y, u, K_{mb}, K_{cb})$ |
| $C \to M$: | $C, B, a, u, y, n_c, T_{cmb}$ |
| $M \to B$: | $M, C, a, u, t, y, n_c, T_{cmb}$ |
| $B$: | checks $T_{cmb}$ |
| | calculates $H(n_c, b_c, K_{cb})$ |
| $B \to M$: | $H(n_c, b_c, K_{cb})$ |
| | transfers the money of amount $u$ |
| $M$: | calculates $T_{bmc} := H(t, u, K_{cm}, H(n_c, b_c, K_{cb}))$ |
| $M \to C$: | $T_{bmc}$ |
| $C$: | checks $T_{bmc}$ |

## 3.4 The secret-revocation procedure

When a customer wants to change her secret, she has to send previously used information with a new secret. We suggest a way of revocation and update of secrets: the customer sends a nonce $n_b$ with $C$ to the bank, then the bank sends back $T_{bc} := H(n_b, K_{cb})$. The customer checks $T_{bc}$, if it is valid, she requests

[1] The secure connection is not necessarily required, since we have the secrets $K_{cb}$ and $K_{mb}$. If we have a secure link between the customer and the merchant, the transaction id $t$ also can be used as a shared secret.

[2] If we are to consider adopting this scheme into existing systems, it is during this money transfer stage that we would overlay our mechanisms. There is usually an intermediate credit card company and the merchant's bank between the customer's bank and the merchant. Transaction clearing is done via this path. There are two typical authentication systems for clearing: one uses signature mechanisms between banks and credit card companies, the other uses secret key mechanisms between them. In both cases, such an infrastructure is already used for banks and credit card companies, but merchants and customers are not in their network. If they extend the infrastructure to all customers and merchants, the cost will be enormous and such extension may not be straightforward. If our scheme is adopted within their infrastructure, no additional investment is needed.

to update her shared information by sending $(C, H(n_b, K_{cb}, b_c) \oplus K'_{cb}, H(K'_{cb}, n_b))$, where $K'_{cb}$ is the new secret. It the customer wants to change $p$ as well, she can do it in the same procedure, but does not have to. When $H(n_b, K_{cb}, b_c)$ is correct, the bank changes the customer's information. Similarly, the customer can change her information in a merchant by using $T_{mc} := H(n_m, K_{cm})$ and $(C, H(n_m, K_{cm}, m_c) \oplus K'_{cm}, H(K'_{cm}, n_m))$, where $n_m$ is a nonce and $K'_{cm}$ is the new secret. If the customer cannot remember her previous information, the information cannot be revoked on-line, and the revocation should be done in off-line communication with both the bank and the merchant.

**Protocol**

$\boxed{C \Longleftrightarrow B}$

| | |
|---|---|
| $C \rightarrow B$: | $C, n_b$ |
| $B \rightarrow C$: | $T_{bc} := H(n_b, K_{cb})$ |
| $C$: | checks $T_{bc}$<br>generates the new secret $K'_{cb}$ |
| $C \rightarrow B$: | $C, H(n_b, K_{cb}, b_c) \oplus K'_{cb}, H(K'_{cb}, n_b)$ |
| $B$: | checks $H(n_b, K_{cb}, b_c)$ and $H(K'_{cb}, n_b)$<br>updates $C$'s secret |
| $B \rightarrow C$: | *ack* |

$\boxed{C \Longleftrightarrow M}$

| | |
|---|---|
| $C \rightarrow M$: | $C, n_m$ |
| $M \rightarrow C$: | $T_{mc} := H(n_m, K_{cm})$ |
| $C$: | checks $T_{mc}$<br>generates the new secret $K'_{cm}$ |
| $C \rightarrow M$: | $C, H(n_m, K_{cm}, m_c) \oplus K'_{cm}, H(K'_{cm}, n_m)$ |
| $M$: | checks $H(n_m, K_{cm}, m_c)$ and $H(K'_{cm}, n_m)$<br>updates $C$'s secret |
| $M \rightarrow C$: | *ack* |

## 4 Discussion

In general, PIN codes are generated by banks, and passwords for services are maintained by service providers though they are generated by service users. Personal secrets like PIN codes are not protected by separation of duty, and this creates a security gap. We showed how to transfer responsibility for secrets to customers in a practical system. Customers generate and maintain their personal secret to access a service. As a result, the service provider is less exposed to the maintenance of customers' secret data. There are no critical customer secrets on the service provider's side.

To minimise communication costs, off-line transactions are preferred in some cases. To make transactions off-line, one has to be able to postpone procedures between the merchant and the bank for some time, and the proof of purchase has to be issued to the customer when the purchase is made. After receiving $T_{cmb}$ from the customer, the merchant can issue $H(t, u, K_{cm}, T_{cmb})$ as the proof of purchase, and the transaction between the customer and the merchant is done. In this case, the merchant does not have a guarantee from the bank, since there is no communication with the bank. As in current off-line transactions using a cheque and cheque guarantee card, we can set a certain credit limit for such transactions and guarantee the merchant his money. The merchant cannot request a different amount of money to bank from the amount in $T_{cmb}$, since $T_{cmb}$ contains $K_{cb}$, and both the customer and the bank will have a hash quantum containing the amount $u$. The customer is also protected.

Existing electronic banking systems keep a file of merchant-customer tuples for authorised transactions. Thus the storage of $T_{cmb}$ will not be a problem in practice. Indeed, limiting the transaction beneficiaries is an important control of fraud, and desirable in the case of off-line merchant transactions. In effect, our scheme allows new merchants to be registered quickly, but only provided the customer, the merchant, and the bank are all on-line at once.

As an alternative of hash functions for our scheme, we could use Message Authentication Codes (MACs). During the registration procedure, principals share secrets $K_{cb}$ between the customer and the bank, and $K_{cm}$ between the customer and the merchant. When we adopt a MAC for the scheme, these secrets can be used as keys for the MAC without further effort. By the nature of MACs, hashed values can be seen only by key sharers. But it does not help reduce the complexity of the scheme.

This scheme provides non-repudiation of both ordering and payment. A merchant cannot make a false transaction without a customer's participation, because he does not have the customer's secret such as $K_{mb}$ and $K_{cb}$. Since a customer has to send a merchant the hash $T_{cmb}$ of a payment id and her private information registered in the bank, the customer cannot deny that the transaction was done by her. For both principals, a merchant and a customer, they cannot forge a transaction by denial for the transaction.

One of the underlying ideas is a customer-oriented transaction; a customer plays a central role in setting up relations to a bank and merchants and making transactions, and keeps her secrets against other principals. Collusion between a bank and a merchant is not so helpful in this model.

## 5 Summary and Future Work

We provided a scalable and resilient access control scheme based only on hash functions, presented applications to electronic payment, and gave a discussion of the scheme. Since this scheme does not share the customer's secret with either the bank or the merchant, we can develop privacy-enhanced and customer-oriented transactions. As the scheme does not deploy a large public key infrastructure, the implementation cost can be lower than conventional schemes such as SET [VM97]. Our scheme can be applied to the existing payment infrastructure with a minimum of change.

We consider applications of our scheme. Paper-based payments using a cheque are still popular, but expensive for banks to process. An on-line electronic cheque can be a system that transfers funds from the payer's bank account to the payee's bank account at the time the transaction takes place. Our scheme is also applicable here, by replacing the merchant with the payee with suitable modifications. One can also consider sophisticated off-line features such as deferred payment, bouncing cheques, etc. Electronic cheque transactions are a topic of future work.

## Acknowledgements

I am grateful to Ross J. Anderson, Václav Matyáš Jr. and the anonymous referees for their helpful comments and advice.

## References

[ALN97] M. Abadi, T. M. A. Lomas, and R. M. Needham. Strengthening passwords. SRC Technical Note 1997-033, Digital Systems Research Center, Palo Alto, CA, September 1997.

[And92] R. J. Anderson. UEPS - a second generation electronic wallet. In *ESORICS '92*, volume 648 of *LNCS*, pages 411–418. Springer-Verlag, 1992.

[And94] R. J. Anderson. Why cryptosystems fail. *Communications of the ACM*, 37(11):32–40, 1994.

[BM92] S. M. Bellovin and M. Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. In *the 1992 IEEE Symposium on Security and Privacy*, pages 72–84, Oakland, CA, 1992.

[BM93] S. M. Bellovin and M. Merritt. Augmented encrypted key exchange: a password-based protocols secure against dictionary attacks and password file compromise. In *the First ACM Conference on Computer and Communications Security*, pages 244–250. ACM SIGSAC, November 1993.

[DA97] T. Dierks and C. Allen. The TLS protocol version 1.0. Internet-DRAFT, Internet Engineering Task Force, November 1997. `<ftp://ftp.ietf.org/internet-drafts/draft-ietf-tls-protocol-05.txt>`.

[GLNS93] L. Gong, T. M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, June 1993.

[Hal94] N. M. Haller. The S/Key one-time password system. In *ISOC Symosium on Network and Distributed System Security*, pages 151–157, San Diego, CA, February 1994. see also IETF RFC 1704, 1760, and 1938.

[MTvHZ92] R. Molva, G. Tsudik, E. van Herreweghen, and S. Zatti. Kryptoknight authenication and key distribution system. In *ESORICS '92*, volume 648 of *LNCS*, pages 155–174. Springer-Verlag, 1992.

[Nee97] R. M. Needham. The changing environment for security protocols. *IEEE Network*, pages 12–15, May/June 1997.

[NS78] R. M. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[NT94] B. C. Neuman and T. Ts'o. An authentication service for computer networks. *IEEE Communications Magazine*, 32(9):33–38, September 1994.

[Rub96] A. D. Rubin. Independent one-time passwords. *USENIX Computing Systems*, 9(1):15–27, 1996.

[VM97] VISA and MasterCard. *SET Secure Electronic Transaction Specification Formal Protocol Definition 1.0*, May 1997.

[Ylö96] T. Ylönen. SSH – secure login connection over thr internet. In *the 6th USENIX UNIX Security Symposium*, pages 37–42, San Jose, CA, June 1996.

# Trusting Trusted Hardware: Towards a Formal Model for Programmable Secure Coprocessors

Sean W. Smith Vernon Austel
*Secure Systems and Smart Cards*
*IBM T.J. Watson Research Center*
*Yorktown Heights, NY 10598-0704*
sean@watson.ibm.com austel@watson.ibm.com

## Abstract

Secure coprocessors provide a foundation for many exciting electronic commerce applications, as previous work [20, 21] has demonstrated. As our recent work [6, 13, 14] has explored, building a high-end secure coprocessor that can be easily programmed and deployed by a wide range of third parties can be an important step toward realizing this promise. But this step requires *trusting* trusted hardware—and achieving this trust can be difficult in the face of a problem and solution space that can be surprisingly complex and subtle.

Formal methods provide one means to express, verify, and analyze such solutions (and would be required for such a solution to be certified at FIPS 140-1 Level 4). This paper discusses our current efforts to apply these principles to the architecture of our secure coprocessor. We present formal statements of the security goals our architecture needs to provide; we argue for *correctness* by enumerating the architectural properties from which these goals can be proven; we argue for *conciseness* by showing how eliminating properties causes the goals to fail; but we discuss how simpler versions of the architecture can satisfy weaker security goals.

We view this work as the beginning of developing formal models to address the trust challenges arising from using trusted hardware for electronic commerce.

## 1 Introduction

**Motivation and Context** Programmable secure coprocessors can be the foundation for many e-commerce applications, as has been demonstrated in the laboratory (e.g., Tygar and Yee's seminal work [20, 21]). To enable real-world deployment of such applications, our team at IBM has recently completed a multi-year project to design and build such a device, that also meets the requirements of realistic manufacturing and use scenarios [13] (and which reached market [6] in the fall of 1997). Bootstrap software that enables secure configuration and programmability, while requiring neither on-site security officers nor trusted shipping channels, played a key part in meeting these requirements. [14]

However, using trusted hardware as a foundation for real e-commerce applications gives rise to an additional requirement: validating that trust. Many e-commerce efforts cite FIPS 140-1 [11] (even though this U.S. government standard addresses crypto modules, not general-purpose secure coprocessors). Certification to this standard means that the device passed a suite of specific tests [4] administered by an independent laboratory. Level 4 (the most stringent level in the standard) requires that secrets are rendered unavailable in all foreseeable physical attacks—the standard definition of a high-end "secure coprocessor." As of this writing, no device has ever been certified at that level.

Our team plans to meet this trust requirement by submitting our hardware for full certification at FIPS 140-1 Level 4, and (as a research project) carrying out the formal modeling and verification that would be necessary for also certifying the bootstrap software at Level 4. Besides applying formal methods to a sizable, implemented system, this project is challenging because it applies a standard originally crafted for cryptographic modules to programmable secure coprocessors, the flexible platform necessary for e-commerce applications.

This effort involves:

- building a formal model of the configuration of a programmable secure coprocessor
- expressing the security requirements in terms of this model
- translating the bootstrap software and other aspects of system behavior into a set of transformations on this model
- formally verifying that the security properties are preserved by these transformations

This paper presents our initial plan of attack: translating into formal methods the goals and requirements that guided the implementation of our device. We have proceeded to carry out this plan, by building the model, assertions, transformations, and proofs within the ACL2 [7] mechanical theorem prover. This work is nearly complete; follow-up reports will detail our experiences with this modeling, and with the FIPS process.

**The Potential of Trusted Hardware** The security of electronic commerce applications requires that participating computers and storage devices correctly carry out their tasks. An adversary who modifies or clones these devices can undermine the entire system. However, in many current and proposed electronic commerce scenarios, the adversary may have physical access to machines whose compromise benefits him. These threats include insider attack, attacks on exposed machines, and dishonest users attacking their own machines.

A *secure coprocessor* is a general-purpose computing device with *secure memory* that is protected by physical and/or electrical techniques intended to ensure that all foreseeable physical attack *zeroizes* its contents. Previous work at our laboratory [12, 16, 17, 18] explored the potential of building a coprocessor with a high-end computational environment and cryptographic accelerators, encapsulated in protective membrane and protected by tamper-response circuitry that senses any physical changes to this membrane. Subsequent work at CMU [15, 20] used these prototypes to define and demonstrate the power of the secure coprocessing model—achieving broad protocol security by amplifying the security of a device trusted to keep its secrets despite physical attack. In particular, this model addresses many security problems in electronic commerce applications [21].

**Building Trusted Hardware** In order to move the secure coprocessing model from the research laboratory into real-world e-commerce applications, our group, over the last two years, has been researching, designing, and implementing a mass-produced high-end secure coprocessor. We needed to accommodate many constraints:

- A device must detect and respond to all foreseeable avenues of tamper.
- We cannot rely on physical inspection to distinguish a tampered device from an untampered device.
- A device must be tamper-protected for life, from the time it leaves the factory—the field is a hostile environment.
- Nearly all software on the device—including most bootstrap, and the operating system—must be remotely replaceable in the field *without security officers, secure couriers, or other trusted physical channels.*
- The different layers of software within any one device must be controllable by different, mutually suspicious authorities—so we need to allow for malicious authorities, as well as *Byzantine failure* of fundamental code (such as the OS or bootstrap).
- Different end-users of the same application (but on different coprocessors) may not necessarily trust each other—and possibly may never have met.
- Nevertheless, an application running on an untampered device must be able to prove, to all concerned, that it's the real thing, doing the right thing.

These constraints were driven by the goal of building a mass-produced tool that enables widespread development of e-commerce applications on secure hardware, much as the earlier generation laboratory prototypes enabled Tygar and Yee's development of research proofs-of-concept [21]. Separate reports discuss the broader problem space [13] and the details of our solution [14] for the resulting commercial product [6].

**Trusting Trusted Hardware** Clearly, a critical component of such a device is effective tamper response: physical attack must zeroize the contents of secure memory, near-instantly and with extremely high probability. Our protections are based on always-active circuitry that *crowbars* memory when it senses changes to a tamper-detecting membrane, which is interleaved with tamper-resistant material to force attacks to affect the membrane. As widely reported (e.g., [1, 2, 19]), designing effective tamper-response techniques and verifying they work can be tricky; [14] discusses the suite of techniques we use in our device, currently under independent evaluation against the FIPS 140-1 Level 4 criteria [4, 11].

Other important components include the hardware and software design that enable fast crypto performance, and the software programming environment that enables applications to easily exploit these abilities.

However, a *security architecture* must connect these pieces in order to ensure that the tamper response actually did any good. Since developing our architecture required simultaneously addressing several complex issues, we find that presenting this material often leads to the same questions:

- "What does a secure coprocessor *do*?"
- "What is your architecture trying to achieve?"
- "Why does it have so many pieces?"
- "Why should I believe it works?"
- "What if I'm solving a simpler problem, and can eliminate property $X$?"

As part of verification of our commercial design, as well as preparation for future work in secure commerce systems, we are currently developing formal models (based on the formalism which guided the implementation) in which to frame and answer these questions. This paper presents some preliminary results. Section 2 presents English statements of some of the overall security goals for a programmable high-end secure coprocessor. Section 3 refines these statements in more formal terms. Section 4 enumerates the elements of our architecture that makes these properties hold. Section 5 presents some arguments for conciseness of design, by showing how eliminating elements of the design causes these properties to fail. But Section 6 shows how weakening the security properties can lead to simpler designs.

This paper presents a snapshot of one aspect of the broader efforts required to ensure that trusted hardware can be trusted. Section 7 explores some of this broader picture.

## 2 English Statements of Security Goals

### 2.1 System Components

As noted earlier, a generic secure coprocessor consists of a CPU, secure memory (which may be both volatile and non-volatile), potentially some additional non-secure non-volatile memory, and often some cryptographic accelerators. (See Figure 1.) All this is packaged in a physical security design intended to render unavailable the contents of secure memory, upon physical attack. Since exploring the effectiveness of physical tamper response lies beyond the scope of this paper, this analysis simply assumes that the physical tamper response works. However, we stress that as part of meeting the trust requirement of trusted hardware, our device's physical security design is undergoing the extensive independent tests required by FIPS 140-1 Level 4.

In our design, the non-volatile secure memory consists of battery-backed static RAM (*BBRAM*). Hardware constraints limited the amount of BBRAM to 8.5 kilobytes; the business goal required that the device carry within it its own software; and our security model did not require that this software be secret[1] from adversaries. Consequently, our design includes a megabyte of FLASH memory[2] as the

[1] Clearly, this design extends to a model where software is secret, by putting in FLASH a two-part program: one part uses secrets in BBRAM to verify and decrypt the second part.

[2] FLASH memory provides rewritable non-volatile storage; but rewriting involves a lengthy process of erasing and mod-

primary code store; this memory is contained with the secure boundary (so attacks on it should first trigger tamper response) but is not itself zeroized upon attack.

The constraints we faced caused us to partition the code-space in FLASH into three layers: a foundational *Miniboot 1* layer, an operating system layer, and an application layer. We refer to a generic layer as *Layer N*; boot-block ROM code is Layer 0, Miniboot 0. See Figure 2.

For each device, each of these layers may potentially be controlled by a different authority. Articulating and accommodating the nature of this control constituted one of the major challenges of moving our secure coprocessor from a laboratory prototype to a real-world device.

- We had to permit the authority to be at a remote site, not where the card is.
- We had to recognize that, potentially, no one at the card's site may be trustworthy.
- We had to allow different authorities to distrust each other.

For a particular device, we refer to the party in charge of the software in Layer $N$ as *Authority N*. Each layer includes code, as well as space for a public key (of the authority over that layer), and some additional parameters.

We partition the BBRAM into a region for each code layer: *Page N* belongs to Layer $N$. Given the multitude of potential owners and the potential failure properties of FLASH, we also must include some notion of the *status* of a code layer at any particular time: e.g., "unowned," "owned but unreliable," "reliable but unrunnable," "runnable."

## 2.2 Desired Properties

Besides accommodating the constraints dictated by business and engineering concerns, our device must make it easy for third party programmers to develop and deploy applications in the style of [21]. This led us to articulate some basic goals for our security architecture.

### 2.2.1 Control of Software

Suppose $A$ has ownership of a particular software layer in particular untampered device. Then only $A$, or a designated superior, can load code into that layer in that device—even though the card may be in a hostile environment, far away from $A$ and her superiors.

### 2.2.2 Access to Secrets

The secrets belonging to layer $N$ are accessible only by code that Authority $N$ trusts, executing on an untampered device that Authority $N$ trusts, in the appropriate context.

### 2.2.3 Authenticated Execution

It must be possible to distinguish remotely between

- a message from a particular layer $N$ with particular software environment on a particular untampered device
- a message from a clever adversary

However, the adversary could have access to other untampered devices, or even to this one, untampered but with a different software environment.

### 2.2.4 Recoverability

We must be able to recover from arbitrary failure in any rewritable layer—including the operating system and rewritable Miniboot.

These failures include:

- the layer contents themselves become scrambled
- the program in that layer behaves with arbitrary evil intent

Furthermore, for this recovery to be meaningful, we must be able to confirm that the recovery has taken place.

ifying a *sector* that is tens of kilobytes long, and can only be done a relatively small number of times compared to ordinary RAM.

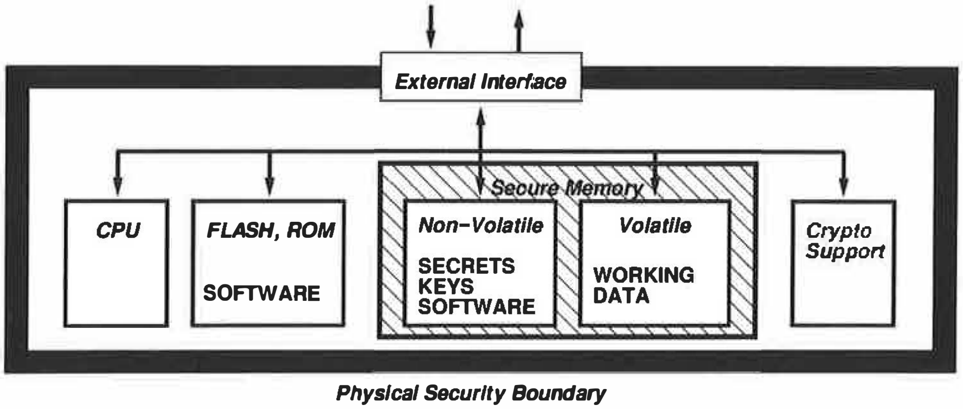


Figure 1: High-level sketch of our secure coprocessor.

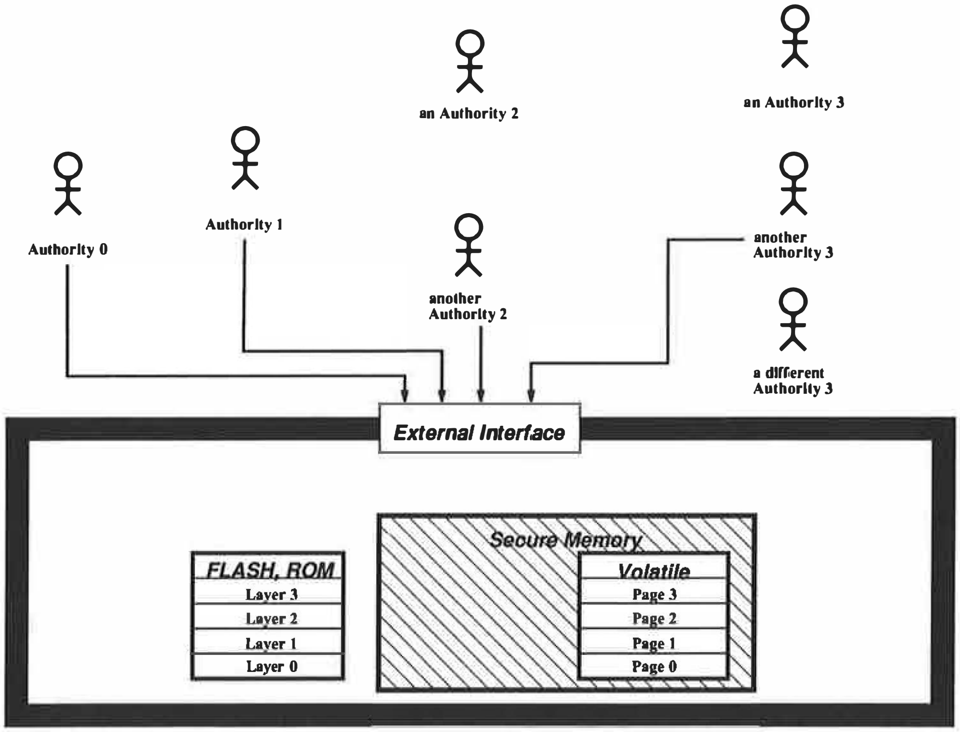


Figure 2: Each code layer has its own subset of BBRAM, and is potentially controlled by a different authority. But different devices may have different authority sets, and the set for any one device may change over time.

### 2.2.5 Fault Tolerance

Interruption of execution at any point will not leave the device in a condition that violates the above.

## 2.3 Justification

Although driven by our particular design constraints, these goals might arguably apply to any general-purpose secure coprocessor intended for third party use, and that is rich enough to require code maintenance.

- (Section 2.2.1) A party who deploys an e-commerce application on such a device wants to be sure that others cannot alter his code. Otherwise, the application can be subverted by an adversary "updating" the program to include a backdoor or Trojan horse.
- (Section 2.2.2) The coprocessor must really provide a trusted environment to safely store application secrets. For example, an adversarial application should not be be able to copy or use secret keys, or modify the balance in an e-wallet.
- (Section 2.2.3) It must be possible for a participant in a coprocessor-based e-commerce application to verify that they are interacting with the correct software on an untampered device. Failure of either of these conditions leaves even basic coprocessor applications, such as cryptographic accelerators, open to attack (e.g., [22]).
- (Section 2.2.4) Failures and interruptions should leave us with at least a safe state, if not a recoverable one; otherwise, an adversary can be expected to cause the necessary failures and interruptions. The cost of the device, and the inevitability of errors in complex software, stress recovery from Byzantine action by loaded software.

Although arguable necessary, whether this set of properties is *sufficient* is an interesting avenue for further exploration.

# 3 Formal Statements of Security Goals

## 3.1 Model Components

Formalizing these English statements requires introducing some terms.

*Layer* $N$, *Authority* $N$, and *Page* $N$ were already introduced in Section 2.1.

The *system configuration* consists of a tuple of the relevant properties, including:

- a vector of conditions for each layer: its status, owner, code contents, and BBRAM page contents
- whether or not the device has been tampered.
- what other hardware failures may have occurred (e.g., does the DRAM scratch area still work?)

We use standard notation $\pi_N C$ to denote the Layer $N$ component of a configuration $C$.

We organize the possible values of these layer components into a natural partial order. For any $A$, "owned by $A$ but unreliable" dominates "unowned." For any $A$ and $P$, "owned by $A$ with reliable and runnable contents $P$" dominates "owned by $A$ with reliable but unrunnable contents $P$," which in turn dominates "owned by $A$ but unreliable."

For each BBRAM page, we need some notion of "initial contents."

For a runnable program at layer $N$, the *software environment of* $N$ in configuration $C$ consists of the programs and status of the components $\pi_K C$ for $0 \leq K \leq N$. We denote this by $ENV_N C$. Our intention here is that the correct and secure execution of the layer $N$ code on an untampered device depends only on the correct and secure execution of the code in its software environment.

**Initial State.** In our design, devices are *initialized* at the factory and left in a configuration where only Layers 0 and 1 are installed, and each of these has a self-generated random secret in its BBRAM page, with a corresponding certificate stored in the FLASH layer. For Layer 1, this is just an RSA keypair, with public key signed by the Factory Certificate Authority ($CA$); for Layer 0 (which, as ROM, does not contain public-key code), this consists of a set of DES keys, and a certificate consisting of the encryption and signature of these keys by the privileged Authority 0.

**Transitions.** The configuration of a device can change due to several potential causes:

- explicit configuration-changing commands, which (in our design) are carried out by the Miniboot layers;
- failures of hardware components, BBRAM and FLASH (discussed further in Section 3.2.5 below)
- tamper events
- ordinary operation of the device, and execution of the code layers.

We model this notion by defining the set of *valid transitions* on configurations, following the above causes. A *valid configuration* is one that can be reached from a valid initial configuration by a sequence of zero or more valid transitions. We frequently discuss a sequence of configurations $C_0 \ldots$ with $C_0$ being some particular valid configuration and with each $C_{i+1}$ reachable from $C_i$ by a valid transition.

**Tamper.** With our earlier assumption that the physical tamper response actually works, perhaps the most natural characterization of tamper is "the device doesn't have its secrets any more." However, this characterization of the effect of tamper-response on secrets overlooks some critical issues:

- The device's secrets may have been copied off-card before tamper, due to exploitation of some error in memory management, and be restored later. (For example, the OS might have a bug that permits a hostile application to download the entire contents of secure memory.)
- The secrets belonging to *another* device may be loaded into this one, after tamper.
- The device's secrets may still be present after tamper, perhaps due a malicious code-loading command that tricked the device into thinking that the secret storage area contained the new code to be burned into FLASH.

As far as secrets go, the only thing we can say for certain is that a tamper event transforms the device configuration by destroying the contents of BBRAM. Immediately after the event, the secrets are no longer in that location.

However, the potential for tamper caused further headaches for formal analysis. The possible configuration transformations for a device are governed by its current code layers, and the physical construction of the device. (Indeed, it is the physical construction which causes the tamper response to occur.) Physical tamper can change these properties—and thus arbitrarily change the possible transformations and behaviors after tamper.

Clearly, reasoning about whether a remote party can authenticate an untampered device requires reasoning about tamper. However, our initial work also found it necessary to include "untamperedness" as an assumption for many other properties—since without it, no memory restrictions or other useful behaviors can hold with any certainty. This is the reason we ended up including "memory of being tampered" as an explicit element in the system configuration: to distinguish traces that have ventured into this terrain, from traces that have remained safe.

**Authentication.** We need to consider scenarios where needs to determine whether a particular message $M$ came from a particular $A$, or from someone else, and we consider authentication schemes based on some secret possessed by $A$.

For such secret-based schemes, we define the *necessary trust set* to consist of those parties who, if they abuse or distribute their secrets, can make it impossible for $B$ to make this distinction. For example, in a standard public-key scheme with a single CA, the necessary trust set would include both $A$, as well as the root CA. (In a scheme with multiple certificate authorities, the trust set would include the intermediate certificate authorities as well.)

### 3.2 Goals

We now attempt to express the goals of Section 2 more formally. (Recall that Section 3.1 discussed the initial conditions of the device.)

#### 3.2.1 Control of Software

**Formal Statement** Suppose:

- an untampered device is in valid configuration $C$
- $N \geq 0$
- $AUTH$ be the set of authorities over $\pi_K C$, for $0 \leq K \leq N$.
- a sequence of valid transitions takes $C$ to some $C'$, where the device is still untampered

If $\pi_k C'$ does not precede or equal $\pi_k C$ in the layer partial order, then:

- at least one transition in the sequence from $C$ to $C'$ was caused by a configuration-changing command
- at least the first such command in this sequence required knowledge of a secret belonging to a member of $AUTH$

**English** An authority must be able to control his layer, and (under appropriate conditions) a superior authority should be able to repair that layer.

However, we also need to recognize the fact that failures of hardware or "trust invariants" also affect a layer's configuration, even without the action of any of these authorities.

The formal statement above attempts to express that one's layer can be *demoted*, due to failure or other reasons, but any change that otherwise modifies the contents must be traceable to an action—now or later—by some *current* Authority $K \leq N$.

#### 3.2.2 Access to Secrets

**Formal Statement** Suppose:

- $A$ is the authority over a layer $N$ in some untampered device in valid configuration $C_0$
- In $\pi_N C_0$, Page $N$ has its initial contents, but no program in $ENV_N C_0$ has yet executed since these contents have been initialized.
- $T_A$ is the set of software environments for $N$ that $A$ trusts.

Let $C_0, ..., C_i, ...$ be a valid sequence of configurations, and let $C_k$ be the first in this sequence such that one of the following is true:

- The device is tampered in $C_k$
- Some $\pi_M C_k$, for $M \leq N$, is not runnable
- $ENV_N C_k \notin T_A$

Then:

- The contents of Page $N$ are destroyed or returned to their initial state in $C_k$.
- For all $0 \leq i < k$, only the programs in $ENV_N C_i$ can directly access page $N$.

**English** This formal statement expresses that once an authority's program establishes secrets, the device maintains them only while the supporting environment is trusted by that authority—and even during this period, protects them from lower-privileged layers.

The overall motivation here is that authority $A$ should be able to trust that any future from configuration $C_0$ is safe, without trusting anything more than $ENV_N C_0$—his environment right now.

In particular, we should permit $A$ to regard any of the following to cause $ENV_N$ to stop being trustworthy:

- Some $C_i$ to $C_{i+1}$ transition might include a change to the code in layer $K < N$, which $A$ does not trust. (Your parent might do something you don't trust.)
- Some $\pi_K C_i$, for $K > N$ and $i \geq 0$, might try to attack layer $N$. (Your child might be try to usurp your secrets.)

- Some other authority in any $C_i$, $i \geq 0$, might behave clumsily or maliciously with his or her private key—e.g., to try to change what the device believes is $A$'s public key. (Your parent might try to usurp your authority.)

### 3.2.3 Authenticated Execution

**Formal Statement** Suppose Bob receives a message $M$ allegedly from layer $N$ on an untampered device with a particular $ENV_N C_k$, for valid configuration $C_k$.

Then Bob can determine whether $M$ came from this program in this environment, or from some adversary, with a necessary trust set as small as possible.

In particular, Bob should be able to correctly authenticate $M$, despite potential adversarial control of any of these:

- new code in any layer in $C_j$, for $j > k$;
- code in any layer in some alternate configuration $C'$ that follows from some $C_j$ via a sequence of valid transformations, one of which is tamper;
- code in any layer $L > N$ in $C_k$;
- old code in any layer $L > 1$ in $C_j$ for $j < k$.

**English** This formal statement basically restates Section 2.2.3, while acknowledging that a clever adversary might load new code into the device, including new bootstrap code, or control code already in a lower-privileged layer in that device, or have controlled code that used to be even in layer $N$, but has since been replaced.

### 3.2.4 Recoverability

**Formal Statement** Suppose:

- an untampered device is in configuration $C$
- $N > 0$,
- $AUTH$ is the set of authorities for 0 to $N - 1$ in $C$.
- $\pi_0 C$ through $\pi_{N-1} C$ are runnable
- $\pi_N C$ is arbitrary (and may have attempted to cause unauthorized configuration changes through its execution)
- $P_N$ specifies proposed new contents for layer $N$.

Then there exists a sequence of configuration-changing commands, from the members of $AUTH$, that takes $C$ to $C'$ where

- $\pi_k C = \pi_k C'$, for $0 \leq k < N$
- $\pi_N C = P_N$.
- the device is still untampered in $C'$
- the members of $AUTH$ can verify that these properties of $C'$ hold

**English** The formal statement basically says that, if layer $N$ is bad but the higher-privileged layers are good, then the authorities over these layers can repair layer $N$.

The final condition expresses a necessary but often-overlooked aspect of code-downloading: the ability to repair a device in a hostile environment is often not meaningful, unless one authenticate that the repair actually took place. Did the new code ever get there? (Any secret used to authenticate this fact must not be accessible to the faulty code.)

### 3.2.5 Fault Tolerance

The above statements asserted properties of untampered devices in a configuration reachable from an initial configuration via a sequence of valid transformations.

To achieve the real-world goal of fault tolerance, it is important that our model be as complete as possible: these transformations must include not just configuration-changing commands and ordinary operation, but also:

- improperly formatted but authentic commands
- early termination of command processing
- as many "hardware failures" (e.g., FLASH storage failures) as possible

Unlike the other desired property, this goal is best expressed as a meta-statement: the other properties hold, even when the model is expanded to cover these conditions.

However, in the process of carrying out the plan outlined in this paper, we discovered that this requirement leads to quite a few subtleties. For example, we did easily extend our initial formal model to express abnormal termination, but we found it inadequate to address improper formatting. (These issues are potentially amenable to formal models, but not with the level of abstraction we chose.) Consequently, we resorted to a separate, systematic code analysis to address improper formatting.

## 3.3 Independence

The initial statements of Section 2 came almost directly from engineering requirements. We note however that, now rendered more formally, the goals appear to remain independent.

For example, "Control of Software" does not imply "Recoverability." A scheme where software could never change, or where software could only change if the public key of that layer's owner was contained in the FLASH segment, would satisfy the former but not the latter.

Conversely, a scheme where *any* layer owner could change layer $N$ establishes that "Recoverability" does not imply "Control of Software."

This remains an area for further investigation, particularly with the mechanical theorem prover. Exploring whether this set of necessary properties could be expressed in a smaller, more concise form could be an interesting problem.

# 4 Correctness

In the previous section, we attempted to render statements of the important security properties in more formal terms. In this section, we now try to examine the assumptions and proof strategies that can verify that the architecture meets these goals.

## 4.1 Architecture Components

Examining why our architecture might achieve its goals first requires articulating some common preliminary subgoals.

### 4.1.1 Boot Sequence

Reasoning about the run-time behavior of the device requires addressing the issue of who is doing what, when.

Our architecture addresses this problem linking device-reset to a *hardware ratchet*:

- a *ratchet* circuit, independent of the main CPU, tracks a "ratchet value" *ratchet*
- hardware ensures that the *reset* signal forces CPU execution to begin in Miniboot 0 bootblock ROM, and forces *ratchet* to zero.
- software on the CPU can increment the ratchet at any point, but only reset can bring it back to zero.

The intention is that each layer $N$ advances the ratchet to $N + 1$ before first passing control or invoking layer $N + 1$; the ratchet then controls access control of code to critical resources.

We model this by introducing an *execution set* of programs that have had control or have been invoked since reset. Hardware reset forces the execution set to empty; invocation or passing control adds a program to the set.

### 4.1.2 Hardware Locks

Who does the hardware permit to access critical memory?

For purposes of this paper, we simplify the behavior of the ratchet to restrict access only to FLASH and BBRAM, according to the following policy:

- The protected FLASH segments can only be written when $ratchet \leq 1$.
- Page $N$ in BBRAM can be read or written only when $ratchet \leq N$.

We stress this is enforced by hardware: the CPU is free to issue a forbidden request, but the memory device will never see it.

### 4.1.3 Key Management

How does the outside world know it's hearing from a real device?

As noted earlier, Miniboot 1 generates a keypair during factory initialization; the private key is retained in Page 1 and the public key is certified by the factory CA. Miniboot 1 can also regenerate its keypair, certifying the new public key with the old private key. In our design, Miniboot 1 generates a keypair for use by Layer 2; the private key is left in Page 2, the public key is certified by Miniboot 1, and the keypair is regenerated each time Layer 2 is altered. Lacking public key code, Miniboot 0 possesses a set of random DES keys used for mutual secret-key authentication with Authority 0.

Section 4.2.3 and Section 5.2 below discuss this in more detail, as does the architecture report [14].

Both the BBRAM and the FLASH locks play a critical role in this solution: the BBRAM locks ensure that the secrets are accessible only by the right layers, and the FLASH locks ensure that the right layers are what we thought they were.

### 4.1.4 Authentication Actually Works

How does the device know it's hearing from the right party in the outside world?

Our design addresses this problem by having each Authority $N$ ($N \geq 1$) generate an RSA keypair and use the private key to sign commands; Miniboot 1 verifies signatures against the public key currently stored in Layer $N$. (With no public-key code in Miniboot 0, Authority 0 uses secret-key authentication.)

However, reasoning that "Miniboot accepted authenticated command from Authority $N$" implies "Authority $N$ signed that command" requires many additional assumptions:

- the intractability assumption underlying cryptography are true;
- keys are really generated randomly;
- the authorized owner of a private key actually keeps it secret;
- Miniboot 0 does not leak its secret keys; and
- Miniboot correctly carries out the cryptographic protocols and algorithms.

### 4.1.5 Code Actually Works

Throughout this work (such as in the last item in the previous section), we continually need to make assertions about "the code actually works."

For example:

- reasoning about the identity of $ENV_N$ in some configuration $C$ requires assuming that the code in previous configurations correctly followed the policy of updating code layers;
- reasoning about the behavior of $ENV_N$ in some configuration requires assuming that $ENV_{N-1}$ correctly evaluated and responded to hardware failures and other status changes affecting Layer $N$, and actually passes control to Layer $N$ when appropriate;
- asserting that only $ENV_N$ can access Page $N$ requires not just the BBRAM locks, but also the assumption that each Layer $K < N$ correctly incremented the ratchet (as well as the assumption that $ENV_N$ is and does what we thought)

Proving these assertions will require careful mutual induction: e.g., establishing that code layers change only through the action of Miniboot requires first establishing that Miniboot hasn't changed in an unauthorized way.

Given the fact that Miniboot 1 itself can change, this means that most trust assertions about the architecture will follow the schema "believing $X$ is true for the system, from now on, requires believing that Miniboot does $Y$ right now."

## 4.2 Assumptions for Goals

### 4.2.1 Control of Software

Establishing that the architecture achieves the "Control of Software" goal requires establishing, as noted above, that only Miniboot, executing as Miniboot can change the FLASH code layers; that authentication of commands works correctly; and then tracing through the possible transformations to show that:

- demotion of status can occur via failure response;
- any other transition requires an authenticated command; and
- the authority issuing this command must have been in the authority set at $C$.

### 4.2.2 Access to Secrets

Establishing that the architecture achieves the "Access to Secrets" goal requires the lemma, noted in Section 4.1.5 above, that only $ENV_N C$ can access Page $N$ in an untampered device, and then showing that all configuration transitions preserve the invariant:

> If $C_i$ differs from $C_{i-1}$ for at least one of the following reasons:
>
> - the device is tampered
> - $ENV_N C_i$ stopped being fully runnable
> - $ENV_N C_i \notin T_A$
>
> then $C_i$ also differs in that the contents of Page $N$ are cleared or returned to initial state.

(Departure from $T_A$ is, from Layer $N$'s perspective, essentially equivalent to tamper.)

Enforcing this invariant requires developing some reasonable way that the device can determine whether or not a new configuration is still a member of $T_A$. We adopted some simple trust parameter schemes to characterize the detectable $T_A$, then further reduce this detectable set by requiring that

- the device itself must be able to confirm that a new $ENV_N \in T_A$
- by *directly* verifying its signature against a reliable public key currently in Layer $N$
- *before* the change occurs

The fact that this invariant is enforced by Miniboot code that is itself part of $ENV_N$, and subject to untrusted change, complicates this implementation. If the $C_k$ transition involved the loading of an untrusted Miniboot, we ensure that trusted Miniboot code currently part of $ENV_N C_{k-1}$ ensures that Page $N$ secrets are erased before the transition succeeds. The fact that, if $A$ does not trust a new Miniboot in $C_k$, it cannot trust Miniboot to correctly carry out authentication in $C_k$ and the future, leads to some additional protocol considerations.

### 4.2.3 Authenticated Execution

Basically, our scheme (Section 4.1.3) binds a public key to certain software environment, and then confines the private key to that environment.

Suppose Bob is trying to authenticate a message from layer $N$, in $C_k$. For simplicity, let suppose $0 \leq N \leq 2$. (The argument extends to the 3-layer schemes discussed in [14].)

The hierarchical nature of code layers, coupled with the fact that each layer $N > 0$ is replaceable, gives a two-dimensional history of code changes. We can extract this history from the configuration sequence $C_0, ...C_i...$ undergone by an untampered device: each $\pi_N C_i$ depends on $\pi_{N-1} C_i$, and is succeeded by $\pi_N C_{i+1}$.

These two dimensions create two dimensions for spoofing: in one axis, some $\pi_j C_k$ might have sent the message, for $j \neq N$; in the other, some $\pi_N C_j$ might have sent the message, for $ENV_N C_j \neq ENV_N C_k$.

If $N = 0$, the preliminary subgoal that only $ENV_N$ sees Page $N$ gives the result—with Bob's necessary trust set consisting of Miniboot 0, $\pi_1 C_0$ (since Miniboot 1 at the factory participates in initialization) and the Authority 0/Factory CA.

If $N = 1$, the preliminary subgoal coupled with the regeneration policy, gives the result—with Bob's necessary trust set consisting of Miniboot 0, the Factory CA, and each version of Miniboot 1 from $\pi_1 C_0$ up to $\pi_1 C_k$. (The fact that any code-changing action of Authority 0 requires repeating factory certification removes Authority 0 from this set).

If $N = 2$, the preliminary subgoal along with the layer 2 key policy gives the result—with Bob's necessary trust set consisting of the $N = 1$ set, along with $\pi_2 C_k$.

This necessary trust set for $N = 1, 2$ is arguably minimal:

- a secret-based scheme forces the factory CA, and $\pi_1 C_0$ (the on-card code that participates in initialization) to be in the set;
- $\pi_N C_k$ must be in the set
- the on-card components in the set must be "connected": some certification path must exist from the initializer $\pi_1 C_0$ to $\pi_N C_k$
- from hierarchical dependence, the set must be "bottom-closed": if $\pi_j C_i$ is in the set, then so must be $\pi_{j-1} C_i$.

### 4.2.4 Recoverability

Establishing Recoverability follows from the subgoals that only $ENV_{N-1}$ sees Page $N - 1$, that only Miniboot changes layers, that permanent Miniboot 0 can correctly replace Miniboot 1 when appropriate, and that the boot sequence always gives Miniboot a chance to authenticate commands. Confirming a successful change follows from the Authenticated Execution property, for Miniboot.

### 4.2.5 Fault Tolerance

Establishing that the architecture meets the Fault Tolerance goal follows from careful code design. Code that already works tests for (and responds to) hardware failures. Configuration transitions need to be structured so that, despite interruptions and other failures, the device is left in a clean, predictable state.

However, various constraints forced our design to depart from standard *atomicity*—a change fails completely or succeeds completely—in a few subtle ways.

First, hardware constraints permit us to have redundant FLASH areas only for Layer 1—so reburns of Layer 2 or 3 force the device first to erase the entire Layer, then reburn it. We handle such destructive transitions by implementing a sequence of two transitions, each of which is atomic. The intermediate failure state is taken to a safe state by the transformation performed by clean-up at the next boot.

Second, some failures can leave the device in a fairly odd state, that requires clean-up by execution of code. We handle these situations by having Miniboot 0 in ROM enforce these clean-up rules, and using the boot sequence subgoal to argue that this clean-up happens before anyone else has a chance to perceive the troubled state. This complicates the formal analysis: atomicity of configuration change does not happen to device configurations *as they exist in real time*, but device configurations *as they can be perceived.*

Our design permits the code authorities to specify what family of untampered devices (and with what software environments) should accept a particular code-loading command. These target features provide the hooks for authorities to enforce the serializability and compatibility rules they find important.

## 5 Conciseness

We argue for conciseness of design by considering two aspects: code loading, and authenticated execution.

### 5.1 Code Loading

Our code-loading involves the "Control of Software," "Access to Secrets" and "Recoverability" goals.

The correctness of our scheme follows from a number of items, include the ratchet locks on FLASH, the ratchet locks on BBRAM, and the trust parameters for what happens to Page $N$ when something in $ENV_N$ is changed.

If our scheme omitted FLASH locks, then we could no longer be sure what layer code is carrying out code changes, and what's in the contents of the code layer that is supposed to be evaluating and carrying out the changes.

If our scheme omitted BBRAM locks, then we would not be able to authenticate whether a change has actually occurred—so, for example, we could never recover from a memory-manager vulnerability in a deployed operating system.

If our scheme forced all code changes to erase *all* BBRAM, then we would lose the ability to authenticate an untampered device while also performing updates remotely in a hostile field—since the device, after the code change, would not be able to prove that it was the same untampered device that existed before the change, one would have to rely on the testimony of the code-loader.

If we did not treat untrusted changes to Miniboot 1 differently from the other layers, then we would force authorities to be in the inconsistent situation of relying on code they no longer trust to correctly evaluate statements about what they do or do not trust.

### 5.2 Authenticated Execution

Our outgoing authentication scheme (Section 4.1.3) forces recipients of a message allegedly from some untampered $\pi_N C_k$ to trust $ENV_N C_k$, $\pi_i C_1$ (for $0 \leq i \leq k$), and the Factory CA. But, as noted earlier, this trust set is arguably minimal: and if any party in this set published or abused its secret keys, then, with this standard PKI approach to authentication the recipient could *never* ascertain whether or not the message came from $\pi_N C_k$.

If the device Page 1 keypair was not regenerated as an atomic part of Miniboot 1 reloads, then it would be possible for new code in $\pi_1 C_{k+1}$ to forge this message. Regeneration protects against attack by future, evil versions of Miniboot 1.

If we did not have a separate keypair for Layer 2, then we'd either have the inefficiency of forcing Layer 2 to reboot the device in order to get a signature (and have Miniboot 1 carefully format what it signs on behalf of Layer 2), or leave the private key outside of Page 1 and have the vulnerability of $\pi_2 C_k$ forging messages from $\pi_1 C_k$.

If we did not regenerate the Layer 2 key with each $ENV_2$ change, then we'd permit $\pi_2 C_{k+1}$ and $\pi_2 C_{k-1}$ to forge messages from $\pi_2 C_k$. (For one example, suppose Authority 2 releases a new operating system to fix a known hole in the old one. Even though the fixed version should retain other operational secrets, not forcing a change of the keypair permits the buggy version to impersonate the fixed version. Conversely, if Authority 2 mistakenly introduces a hole with a new version, not forcing a change of the keypair permits the new version to impersonate the old.)

If we did not have the BBRAM locks, then $\pi_j C_k$ could forge messages from $\pi_N C_k$ for general $j > N$. If we did not have the FLASH locks, then any $\pi_j C_i$ for $0 \leq i \leq k$ might be forging this message.

## 6 Simplifications

We developed our security architecture to support a secure coprocessor with a specific set of constraints. However, the flexibility and security goals for our "high-end" product forced us (for the most part) into a difficult situation. (Our one "easy way out" was the freedom *not* to spontaneously load arbitrary, mutually suspicious peer applications.)

To put it succinctly, our design is an arguably minimal solution to an arguably pessimal problem. Simplifying the problem can certainly simplify the solution. For example:

- Forcing any configuration change to $ENV_N$ to kill Page N—except for the outgoing authentication keys—would simplify enforcement of Access to Secrets.
- Only allowing for one code authority—or for a family of mutually trusting authorities—would simplify authentication and trust.
- If we assume the OS will never have any memory access vulnerabilities, then we could eliminate FLASH locks and BBRAM locks: memory management would be enforced by code we trusted.
- If Miniboot, and possibly the OS, were never to be changed, then the key management and secret-access schemes can be greatly simplified.

We anticipate that many of these simplifications may arise if this architecture were mapped to a smaller device, such as a next-generation smart card.

## 7 Related and Future Work

Much previous and current work (e.g., [3, 7, 9]) explores the use of formal methods as tools to examine the basic question of: *does the design work?* Many recent efforts (e.g., [5, 8, 10]) apply automated tools specifically to electronic commerce protocols.

As noted earlier, this paper reports our initial strategy for formal verification of our existing implementation of an e-commerce tool. Having refined the design goals into statements about a formal model, the next step is to express and verify these properties with a mechanical verification tool—and to, in accordance with FIPS 140-1 Level 4, rigorously document how the model and transformation system correspond to the actual device and its code. This work is underway, as is independent verification of the physical security of our device.

However, we have found that convincing users that our trusted hardware can indeed be trusted also requires addressing additional issues. For example:

- *Does the implementation match the design?* History clearly shows that "secure" software is fraught with unintended vulnerabilities. In our work, we address this with several techniques:
    - evaluation by independent laboratories (e.g., as part of FIPS certification)
    - continual consultation and evaluation by in-house penetration specialists
    - following general principles of careful coding design such as clearing the stack, and safely tolerating input that is deranged (e.g., negative offsets) even if authenticated
- *Does the product, when purchased, match the implementation?* Our policy of "tamper-protection for life" protects the device when it leaves the factory; we have procedures in place to address potential attack before that point.

The existence of flexible, powerful and trusted secure coprocessors can help secure computation in untrusted environments. This research is one part of our group's broader efforts to achieve this security by making these tools available.

## Acknowledgments

The authors gratefully acknowledge the contributions of entire Watson development team, including Dave Baukus, Suresh Chari, Joan Dyer, Gideon Eisenstadter, Bob Gezelter, Juan Gonzalez, Jeff Kravitz, Mark Lindemann, Joe McArthur, Dennis Nagel, Elaine Palmer, Ron Perez, Pankaj Rohatgi, David Toll, Steve Weingart, and Bennet Yee; the IBM Global Security Analysis Lab at Watson, and the IBM development teams in Vimercate, Charlotte, and Poughkeepsie.

We also wish to thank Ran Canetti, Michel Hack, Matthias Kaiserswerth, Mike Matyas, and the referees for their helpful advice, and Bill Arnold, Liam Comerford, Doug Tygar, Steve White, and Bennet Yee for their inspirational pioneering work.

## Availability

**Hardware** Our secure coprocessor exists as the IBM 4758, a commercially available PCI card. (Additional research prototypes exist in in PCMCIA format.)

**Software** Toolkits exist for independent parties to develop, experiment with, and deploy their own applications on this platform. In addition, application software available from IBM transforms the box into a cryptographic accelerator.

**Data** More information—including development manuals—is available on the Web:

`www.ibm.com/security/cryptocards/`

## References

[1] R. Anderson, M. Kuhn. "Tamper Resistance—A Cautionary Note." *The Second USENIX Workshop on Electronic Commerce.* November 1996.

[2] R. Anderson, M. Kuhn. *Low Cost Attacks on Tamper Resistant Devices.* Preprint. 1997.

[3] E. M. Clarke and J. M. Wing. "Formal Methods: State of the Art and Future Directions." *ACM Computing Surveys.* 28: 626-643. December 1996.

[4] W. Havener, R. Medlock, R. Mitchell, R. Walcott. *Derived Test Requirements for FIPS PUB 140-1.* National Institute of Standards and Technology. March 1995.

[5] N. Heintze, J. D. Tygar, J. M. Wing, H. C. Wong. "Model Checking Electronic Commerce Protocols." *The Second USENIX Workshop on Electronic Commerce.* November 1996.

[6] *IBM PCI Cryptographic Coprocessor.* Product Brochure G325-1118. August 1997.

[7] M. Kaufmann and J. S. Moore. "An Industrial Strength Theorem Prover for a Logic Based on Common Lisp." *IEEE Transactions on Software Engineering.* 23, No. 4. April 1997.

[8] D. Kindred and J.M. Wing. "Fast, Automatic Checking of Security Protocols." *The Second USENIX Workshop on Electronic Commerce.* November 1996.

[9] C. Meadows. "Language Generation and Verification in the NRL Protocol Analyzer." *Proceedings of the 9th Computer Security Foundations Workshop.* 1996.

[10] C. Meadows and P. Syverson. "A Formal Specification of Requirements for Payment Transactions in the SET Protocol." *Proceedings of the Second International Conference on Financial Cryptography.* Springer-Verlag LNCS. To appear, 1998.

[11] National Institute of Standards and Technology. *Security Requirements for Cryptographic Modules.* Federal Information Processing Standards Publication 140-1, 1994.

[12] E. R. Palmer. *An Introduction to Citadel—A Secure Crypto Coprocessor for Workstations.* Computer Science Research Report RC 18373, IBM T. J. Watson Research Center. September 1992.

[13] S. W. Smith, E. R. Palmer, S. H. Weingart. "Using a High-Performance, Programmable Secure Coprocessor." *Proceedings of the Second International Conference on Financial Cryptography.* Springer-Verlag LNCS. To appear, 1998.

[14] S. W. Smith, S. H. Weingart. *Building a High-Performance, Programmable Secure Coprocessor.* Resarch Report RC21102. IBM T.J. Watson Research Center. February 1998. (A preliminary version is available as Resarch Report RC21045.)

[15] J. D. Tygar and B. S. Yee. "Dyad: A System for Using Physically Secure Coprocessors." *Proceedings of the Joint Harvard-MIT Workshop on Technological Strategies for the Protection of Intellectual Property in the Network Multimedia Environment.* April 1993.

[16] S. H. Weingart. "Physical Security for the $\mu$ABYSS System." *IEEE Computer Society Conference on Security and Privacy.* 1987.

[17] S. R. White, L. D. Comerford. "ABYSS: A Trusted Architecture for Software Protection." *IEEE Computer Society Conference on Security and Privacy.* 1987.

[18] S. R. White, S. H. Weingart, W. C. Arnold and E. R. Palmer. *Introduction to the Citadel Architecture: Security in Physically Exposed Environments.* Technical Report RC 16672, Distributed Systems Security Group. IBM T. J. Watson Research Center. March 1991.

[19] S. H. Weingart, S. R. White, W. C. Arnold, and G. P. Double. "An Evaluation System for the Physical Security of Computing Systems." *Sixth Annual Computer Security Applications Conference.* 1990.

[20] B. S. Yee. *Using Secure Coprocessors.* Ph.D. thesis. Computer Science Technical Report CMU-CS-94-149, Carnegie Mellon University. May 1994.

[21] B. S. Yee, J. D. Tygar. "Secure Coprocessors in Electronic Commerce Applications." *The First USENIX Workshop on Electronic Commerce.* July 1995.

[22] A. Young and M. Yung. "The Dark Side of Black-Box Cryptography—or—should we trust Capstone?" *CRYPTO 1996.* LNCS 1109.

# On Secure and Pseudonymous Client-Relationships with Multiple Servers

Daniel Bleichenbacher Eran Gabber Phillip B. Gibbons
Yossi Matias* Alain Mayer

*Bell Laboratories, Lucent Technologies*
*600 Mountain Avenue*
*Murray Hill, NJ 07974*
*{bleichen, eran, gibbons, matias, alain}@research.bell-labs.com*

## Abstract

*This paper introduces a cryptographic engine,* Janus, *that assists clients in establishing and maintaining secure and pseudonymous relationships with multiple servers. The setting is such that clients reside on a particular subnet (e.g., corporate intranet, ISP) and the servers reside anywhere on the Internet. The Janus engine allows for each client-server relationship to use either weak or strong authentication on each interaction. At the same time, each interaction preserves privacy by neither revealing a client's true identity ("modulo" the subnet) nor the set of servers with which a particular client interacts. Furthermore, clients do not need any secure long-term memory, enabling scalability and mobility. The interaction model extends to allow servers to send data back to clients via e-mail at a later date. Hence, our results complement the functionality of current network anonymity tools and remailers.*

## 1 Introduction

We consider the following problem: there is a set of clients located on a particular subnet and a set of servers on the Internet. For example, the set of clients could be employees on a company's intranet or subscribers of an ISP and the servers could be Web-sites. See Figure 1, where the $c_i$ are clients and the $s_j$ are servers. A client wishes to establish a persistent relationship with some (or all) of these servers, such that in all subsequent interactions (1) the client can be recognized and (2) either weak or strong authentication can be used. At the same time, clients may not want to reveal their true identity nor enable these servers to determine the set of servers each client has interacted with so far (establishing a dossier). This last property is often called *pseudonymity* to denote persistent anonymity. Equivalently, a client does not want a server to infer through a relationship more than the subnet on which the client is located, nor to connect different relationships to the same client. This paper introduces a client-based cryptographic engine, which allows a client to efficiently and transparently establish and maintain such relationships using a single secret passphrase. Finally, we extend our setting to include the possibility of a server sending data via e-mail to a client.

We consider the specification and construction of a cryptographic function that is designed to assist in obtaining the above goal. Such a function needs to provide a client, given a *single passphrase*, with either a *password* (weak authentication) or a *secret key* (strong authentication) for each relationship. Furthermore, a *username* might be needed as well, by which a client is (publicly) known at a server. Such passwords, secret keys, and usernames should neither reveal the client's true identity nor enable servers to establish a dossier on the client. We name such a cryptographic function (engine) the *Janus* function (engine). We will briefly review arguments why simple choices for the Janus function, such as a collision-resistant hash function, are not quite satisfactory for our purposes and consequently, we will show a Janus function that is more robust. We will also show how to implement a mailbox system on the client side, such that a server can send e-mail to a client without requiring any more information than for client authentication.

*Also with Computer Science Dept., Tel-Aviv University, Tel-Aviv 69978 Israel. E-mail: matias@math.tau.ac.il.

### 1.1 Related Work and Positioning of Our Work

Network anonymity is being extensively studied (see, e.g., [PW85, GWB97]). For example, Simon in [S96] gives a precise definition for an *Anonymous Exchange Protocol* allowing parties to send individual messages to each other anonymously and to reply to a received message. Implementation efforts for approximating anonymous networks are being carried out by several research groups (e.g., anonymous routing [SGR97] and anonymous Web traffic [SGR97, RR98]). Besides that, there are several anonymous remailers available for either e-mail communication (see, e.g., [GWB97, GT96, B96, E96]) or Web browsing (see, e.g., [Anon]). We will discuss some of these in more detail later.

We view our goal as *complementary*: All of the above work tries to find methods and systems to make the Internet an (approximate) anonymous network. This is a hard task and consequently the resulting tools are rather difficult to use and carry some performance penalties. We focus on a method for assisting a client to interact with multiple servers easily and efficiently, such that the server cannot infer the identity of the clients among all clients in a given subnet, but at the same time the client can be recognized and authenticated on repeat visits. We do not address communication between a subnet and a server. Consequently, a server can easily obtain the particular subnet in which a client is located. In many cases, this degree of anonymity is sufficient, for example, if the client is a subscriber of a large ISP, or an employee of a large company. In the language of Reiter and Rubin [RR98], the anonymity of such a client is somewhere between *probable innocence* and *beyond suspicion*. Alternatively, our method can be used in conjunction with existing remailers to enable a client to interact with a server without revealing the particular subnet. We elaborate on this point in Section 2 for client-initiated traffic and in Section 4.3 for server-initiated traffic. The work closest in spirit to the Janus engine are the visionary papers of Chaum [C81, C85] on *digital pseudonyms*.

In [GGMM97], we described the design and implementation of a Web-proxy which assists clients with registering at multiple Web-servers. In this paper, we focus on a new, simpler, and *correct* construction of the Janus engine, a new and different method of conveying anonymous e-mail that greatly reduces the required trust in the intermediary, and a discussion of moving features to shift trust from a proxy to the client's machine. The latter allows, for example, a Janus engine to be integrated with the P3P proposal, giving clients the power to use pseudonymous P3P personae (see Section 5). Thus, our methods and design are applicable to a variety of client-server interactions, well beyond the proxied Web browsing for server registration of [GGMM97].

**Outline:** In Section 2 we describe our interaction model and our function requirements. Section 3 contains a detailed description of the Janus function. Section 4 extends the model of interaction to allow servers to send data to clients' anonymous mailboxes. Finally, Section 5 presents various applications and configurations and discusses some of the trade-offs involved.

## 2 Model and Specifications

In this section, we present the framework for interaction between clients and servers, and the way in which the Janus engine is incorporated within such interaction. There is a set of clients $\mathcal{C} = \{c_1, c_2, \ldots, c_N\}$ and a set of servers $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$. Each client can interact with any server. Interaction can take place in one of the following two ways:

- *Client-initiated*: A client $c_i$ decides to contact a server $s_j$. The server $s_j$ requires $c_i$ to present a username and a password (secret shared key) at the beginning of this interaction to be used for identification and weak (strong) authentication on repeat visits.
- *Server-initiated*: A server $s_j$ decides to send some data to a client $c_i$ which has contacted $s_j$ at some earlier point in time (using the client's username).

Individual clients may wish to remain anonymous in the above interaction; i.e., a client does not want to reveal her real identity $c_i$ to a server (beyond the particular subnet on which $c_i$ is located).

**Client-initiated interaction:** A client $c_i$, on a first visit, presents to a server $s_j$ an *alias* $a_{i,j}$, which includes a username and either a password or a key. On repeat visits a client simply presents the password again for weak authentication or uses the key with a message authentication code (MAC) for strong authentication (see [MMS97]). We would like the alias $a_{i,j}$ to depend on the client $c_i$, the server $s_j$, and a secret client passphrase $p_i$. Since we want this translation of names to be computable, we define a function which takes $c_i$, $p_i$ and $s_j$, and returns

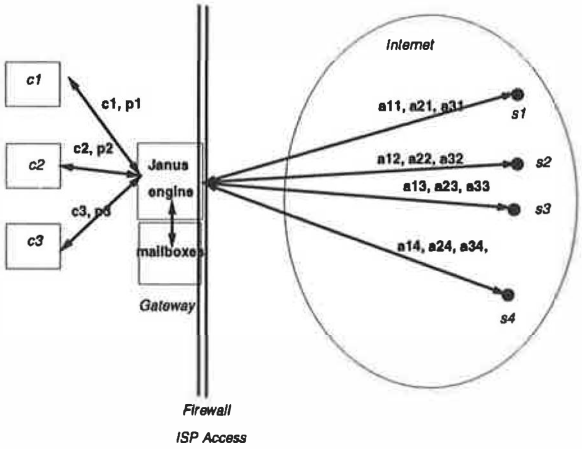


Figure 1: Client Server Configuration with Janus Engine on the Gateway

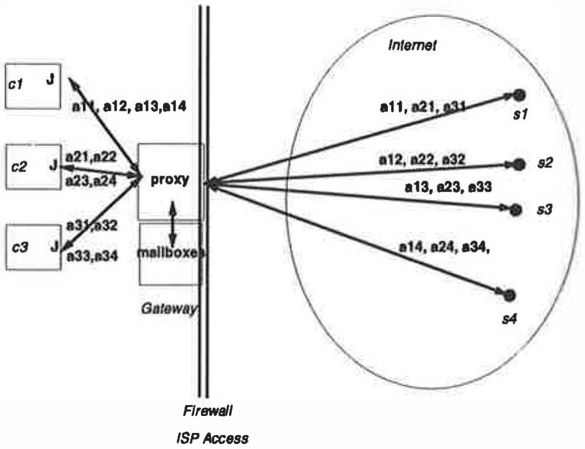


Figure 2: Client Server Configuration with Local Janus Engine

an alias $a_{i,j}$. This function is called the *Janus function*, and is denoted $\mathcal{J}$. In order to be useful in this context, the *Janus function* has to fulfill a number of properties:

1. *Form properties:* For each server, $\mathcal{J}$ provides each client with a *consistent* alias, so that a client, by giving her unique identity and passphrase, can be recognized and authenticated on repeat visits. $\mathcal{J}$ should be *efficiently* computable given $c_i$, $p_i$, and $s_j$. The alias $a_{i,j}$ needs to be accepted by the server, e.g., each of its components must have appropriate length and range.

2. *Secrecy of passwords/keys:* Alias passwords/keys remain secret at all times. In particular, an alias username does not reveal information on any alias password/key.

3. *Uniqueness of aliases among clients & Impersonation resistance:* Given a client's identity and/or her alias username on a server $s_j$ a third party can guess the corresponding password only with negligible probability. Moreover, the distribution of the alias usernames should be such that only with negligible probability two different users have the same alias username on the same server.

4. *Anonymity / Uncheckability of clients:* The identity of the client is kept secret; that is, a server, or a coalition of servers, cannot determine the true identity of the client from her alias(es). Furthermore, it is not checkable whether a particular client is registered at a given server.

5. *Modular security & Protection from creation of dossiers:* An alias of a client for one server does not reveal any information about an alias of the same client for another server. This also implies that a coalition of servers is unable to build a client's profile (dossier) based on the set of servers with which he/she interacted by simply observing and collecting aliases.

One possible physical location to implement the Janus function is on the gateway. See Figure 1, where we refer to the implementation as the *Janus engine*. Clients provide their identity $c_i$ and secret passphrase $p_i$ to the gateway, where the translation takes place. An alternative location for the Janus engine is on each client's machine, as depicted in Figure 2, where the locally generated aliases are sent to the server via the gateway. See Section 5 for a discussion of trade-offs. The following property is of practical significance, as it provides robustness against the possibility to recover privacy-sensitive information "after the fact":

6. *No storage of sensitive data:* When a client is not interacting with a server, the Janus engine does not maintain in memory any information that may compromise on the above properties of the Janus function. This excludes the simple approach of implementing a Janus function by a look-up table.

Consequently, an entity tapping into a (gateway) machine on the subnet, cannot infer any useful information, unless it captures a client's passphrase (which is never transmitted in Figure 2). Additionally, a client can use different Janus engines within her subnet, given that she remembers her passphrase (*mobility*).

If a client desires to hide her subnet from a server, she can easily combine our method with other anonymity tools. For example, if she contacts a server via the Web (HTTP), she can use either Onion Routing [SGR97] or Crowds [RR98]. In the first case, the connection from the gateway to the server is routed and encrypted similar to the methods used by type I/II remailers (see also Section 4.3) and in the second case her connection is "randomly" routed among members (on different subnets) of a crowd.

**Server-initiated interaction:** A server knows a client only by the alias presented in a previous, client-initiated interaction. We allow a server $s_j$ wishing to send data to client $c_i$, known to it as $a_{i,j}$, to send an e-mail message to the corresponding subnet, addressed to the username component $u$ of $a_{i,j}$. The message is received by the Janus engine, see Figure 1, which will make sure that the message is delivered to the appropriate client, or is stored by the gateway, until a local Janus engine retrieves the messages, as in Figure 2. Our scheme of storing mailboxes maintains forward secrecy. More details are in Section 4, where it is also shown how server-initiated interaction can be combined with pseudonymous remailers.

## 3 The Janus Function

In this section we present the Janus function in detail. We first develop our requirements, then discuss some possible constructions.

**The Setting of the Janus-function:** A client inputs her identity $c_i$, her secret passphrase $p_i$, the identity of the server $s_j$, and a tag $t$ indicating the purpose of the resulting value. Depending on this tag, the Janus function returns either an alias-username $a^u_{i,j}$ for the user $c_i$ on the server $s_j$ or the corresponding password $a^p_{i,j}$. In this section we use the two tags $u, p$, but we can easily extend the function by adding additional tags, for generating secret values for other purposes (see also [MMS97]). For example, in Section 4 we extend the Janus function to a third tag, $m$, for the purposes of anonymous mailboxes.

**Adversarial Model:** We assume that a client $c_i$ does not reveal her passphrase $p_i$ to anyone (other than the Janus engine). However, we allow that an adversary $E$ can collect pairs $(a^u_{i,j}, a^p_{i,j})$ and the corresponding server names $s_j$. Note that registered alias usernames may be publicly available on some servers and that we can not assume that all servers can be trusted or store the passwords securely. In some cases it might even be possible to deduce a client name $c_i$ (e.g., from the data exchanged during a session, or simply because the client wishes to disclose her identity) and we also have to assume that a chosen message attack is possible (e.g., by suggesting to a client $c_i$ to register on a specific server). Roughly speaking, we will require that an adversary does not learn more useful information from the Janus function than he would learn if the client would chose all her passphrases and aliases randomly.

### 3.1 Janus function specifications

**Definition 1** *We say that a client $c_i$ is* **corrupted** *if the adversary $E$ has been able to find $p_i$. We say that $c_i$ is* **opened** *with respect to a server $s_j$ if the pair $(a^u_{i,j}, a^p_{i,j})$ has been computed and used. (Note that if $c_i$ has been opened with respect to a server $s_j$ then an adversary $E$ may know only $(a^u_{i,j}, a^p_{i,j})$ but not necessarily $c_i$.) We say that $c_i$ has been* **identifiably opened** *with respect to a server $s_j$ if an adversary knows $(a^u_{i,j}, a^p_{i,j})$ together with the corresponding $c_i$.*

Let $\mathcal{C}$ be the set of clients, $\mathcal{S}$ be the set of servers, $\mathcal{P}$ be the set of allowable client secret passwords, $\mathcal{A_U}$ be the set of allowable alias usernames, and $\mathcal{A_P}$ be the set of allowable alias passwords. Let $k$ be the security parameter of our Janus function meaning that a successful attack requires about $2^k$ operations on average. Let the Janus function be $\mathcal{J} : (\mathcal{C} \times \mathcal{S} \times \mathcal{P} \times \{u, p, m\}) \mapsto \{0,1\}^k$.
Since usernames and passwords normally consist of a restricted set of printable characters we also need two functions that simply convert general $k$-bit strings into an appropriate set of ASCII strings. Thus let $\pi_U : \{0,1\}^k \mapsto \mathcal{A_U}$ and $\pi_P : \{0,1\}^k \mapsto \mathcal{A_P}$ be two injective functions that map $k$-bit strings into the set of allowable usernames and passwords. Let $c_i \in \mathcal{C}$ and $p_i \in \mathcal{P}$. The client's identity $a^u_{i,j}$ and password $a^p_{i,j}$ for the server $s_j$ are then computed by

$$\begin{aligned} a^u_{i,j} &:= \pi_U(\mathcal{J}(c_i, s_j, p_i, u)) \\ a^p_{i,j} &:= \pi_P(\mathcal{J}(c_i, s_j, p_i, p)). \end{aligned}$$

The two functions $\pi_U$ and $\pi_P$ are publicly known, easy to compute and we may assume easy to invert. Thus knowing $\pi_U(x)$ of some $x$ is as good as knowing $x$. In particular if an adversary can guess $\pi_U(x)$ then he can guess $x$ with the same probability.

Following our adversarial model, the Janus function has to satisfy the following requirement:

1. *Secrecy:* Given a server $s_j$, an uncorrupted and not identifiably opened client $c_i$ and $t \in \{p, u, m\}$, the adversary $E$ cannot find $\mathcal{J}(c_i, s_j, p_i, t)$ with nonnegligible probability even under a chosen message attack, that is under the assumption that the adversary can get $\mathcal{J}(c_i, s_{j'}, p_i, t')$ for any $s_{j'} \neq s_j$ or $t \neq t'$.

2. *Anonymity:* Given a server $s_j$, two uncorrupted clients $c_i, c_{i'}$ that are not opened with respect to $s_j$ and $t \in \{p, u\}$. Then an adversary cannot distinguish $\mathcal{J}(c_i, s_j, p_i, t)$ from $\mathcal{J}(c_{i'}, s_j, p_{i'}, t)$ with nonnegligible probability even under a chosen message attack, that is under the assumption that the adversary can get $\mathcal{J}(c_{i''}, s_{j'}, p_{i''}, t')$ for any list of arguments not used above.

Note that the two requirements are indeed different. For example if we were to implement the function $\mathcal{J}$ using a digital signature scheme, i.e., $\mathcal{J}(c_i, s_j, p_i, t) = \mathrm{sig}_{p_i}(c_i||s_j||t)$, then the first requirement would be satisfied, but not the second one, since the client's identity could be found by checking signatures. On the other hand a constant function satisfies the second requirement, but not the first one.

Our requirements are stated in a rather general form. In particular, the first requirement states that no result of the Janus function can be derived from other results. This implies the secrecy of passwords, impersonation resistance and modular security.

## 3.2 Possible Constructions for $\mathcal{J}$

Assume that $\ell_c$ is the maximal bit length of $c_i$, $\ell_s$ the maximal length of $s_j$, $\ell_p$ the maximal length of $p_i$ and $\ell_t$ the number of bits required to encode the tag $t$. Throughout this section we will assume that all inputs $c_i, s_j, p_i$ are padded to their maximal length. This will assure that the string $c_i||s_j||p_i||t$ is not ambiguous.

Given the function specification, an ideal construction would be via a pseudorandom function $f : \{0,1\}^{\ell_c+\ell_s+\ell_p+\ell_t} \mapsto \{0,1\}^k$. Unfortunately, there are no known implementations of pseudorandom functions. Typically, they are approximated via either strong hash functions or message authentication codes (MAC), even though, strictly speaking, the definitions of these primitives do not require them to be pseudorandom. In the following sections, we are going to examine both options and give some justifications for preferring a MAC based solution over other tempting constructions.

### 3.2.1 Using hash functions

One possible attempt might be to use the hash of the inputs $h(c_i||s_j||p_i||t)$ as our function. However, hash functions are not designed to keep their inputs secret. Even if it is hard to invert the hash function for a given input, it might still be possible to derive $p_i$ given $h(c_i||s_j||p_i||t)$ for many different servers $s_j$. A hash function that is weak in that respect can for example be found in [A93]. Some apparently better constructions for keyed functions based on hash functions have been proposed (e.g., MDx-MAC [PO95]). But our requirements are quite different from the goals of these constructions. Therefore, we decided not to use hash functions for our Janus function.

### 3.2.2 MACs

A much more promising approach is the use of message authentication codes (MACs). In particular if $\mathrm{MAC}_K(x)$ denotes the MAC of the message $x$ under the key $K$ then we can define a potential Janus function as

$$\mathcal{J}(c_i, s_j, p_i, t) = \mathrm{MAC}_{p_i}(c_i||s_j||t).$$

This approach has the advantage that some of our requirements are already met. In particular if the MAC is secure then the secrecy of passwords and impersonation resistance for the Janus function are implied. Other requirements, like consistency, efficient computation of the function, single secret and acceptability, are just consequences of the actual implementation of the Janus function and the mappings $\pi_U$ and $\pi_P$. The only additional requirement is the anonymity of clients.

To this end, we consider the following result of Bellare, Kilian, and Rogaway ([BKR94]): Let $x = x_1, \ldots, x_m$ be a message consisting of $m$ blocks $x_i$ of size $\ell$ bits. Given a block cipher $f_K : \{0,1\}^\ell \mapsto \{0,1\}^\ell$ where $K$ denotes the key, define the CBC-MAC by

$$\mathrm{MAC}_K(x) = f_K(\cdots f_K(f_K(x_1) \oplus x_2) \cdots \oplus x_m).$$

Assume that an adversary can distinguish a $\mathrm{MAC}_K$ from a random function with an advantage $\epsilon$ by running an algorithm in time $t$ and making $q$ queries to an oracle that evaluates either $\mathrm{MAC}_K$ or the random function. Then the adversary can distinguish $f_K$ from a random function running an algorithm of about the same size and time complexity having an advantage of $\epsilon - q^2m^2 2^{-\ell-1}$. Hence, if we use CBC-MACs, then anonymity is just a consequence of [BKR94].

If the underlying block cipher $f_K$ behaves like a pseudorandom function then the above result shows that a birthday attack is almost the best possible attack. In particular an attacker can do not much better than collecting outputs of the function and hoping for an internal collision, i.e. two messages $x, y$ such that $f_K(f_K(\cdots f_K(f_K(x_1) \oplus x_2) \cdots \oplus x_{i-1}) \oplus x_i) = f_K(f_K(\cdots f_K(f_K(y_1) \oplus y_2) \cdots \oplus y_{i-1}) \oplus y_i)$ for some $i < m$. In that case the attacker would know that replacing the first $i$ blocks in any message starting with $x_1, \ldots, x_i$ by $y_1, \ldots, y_i$ would result in another message having the same hash value.

We thus caution that a block cipher with $\ell$-bit block size should not be used if an attacker can collect about $2^{\ell/2}m^{-1/2}$ MACs. Concretely, block ciphers having 64-bit blocks, such as DES, triple-DES, or IDEA [LM91] should not be used if it is feasible for an attacker to collect about $2^{32}$ samples, thus giving only marginal security to the overall scheme. However, newer block ciphers, such as SQUARE [DKR97] and one variant of RC5 [R95] have 128-bit block sizes and are therefore more suitable in this case.

# 4 An Anonymous Mailbox System

We will first summarize the history of anonymous remailers, then describe our anonymous mailbox system, and finally discuss how enhanced privacy can be achieved by using our mailbox system in conjunction with remailers.

## 4.1 Brief History of Anonymous E-mail

Tools for anonymous e-mail communication have been around for a few years by now (see, e.g, [GWB97, B96, GT96, E96]. Early anonymous remailers (Type 0, e.g., `Anon.penet.fi`) accepted e-mail messages by a user, translated them to a unique ID and forwarded them to the intended recipient. The recipient could use the ID to reply to the sender of the message. The level of security of this type of remailer was rather low, since it did not use encryption and kept a plain text (translation) database. A next (and still current) generation of remailers (Type I, Cypherpunk remailers) simply take a user's e-mail message, strip off all headers and send it to the intended recipient. The user can furthermore encrypt the message before sending it and the remailer will decrypt the message before processing it. For enhanced security, a user can *chain* such remailers. In order to use a chain $r_1 - r_2$ of remailers, a user first encrypts the message for $r_2$ and then for $r_1$. (see also the efforts on Onion Routing, [SGR97]). Still, even such a scheme is susceptible to traffic analysis, spam and replay attacks. Mixmaster remailers (Type II) are designed to withstand even these elaborate attacks. This development of remailer yields more and more intraceable way of sending messages, but it gives no way to reply to a message. This gives rise to "pseudonymous / nym" remailers, which, in a nutshell, work as follows: A user chooses a pseudonym (nym), which has to be unused (at that remailer). Then the user creates a public/private key pair for that nym. When sending a message, the user encrypts with the server's public key and signs a message with her private key. The recipient can reply to the message using the nym. Some remailers store the message and the original sender can retrieve this mail by sending a signed command to the remailer, other remailers directly forward the message by using a "reply block", an encrypted file with the user's real e-mail.

The ultimate goal of all these remailers is to enable e-mail communication as if the Internet were an anonymous network. This is a very hard task and consequently these tools induce a performance penalty and are rather difficult to use.

## 4.2 Anonymous Mailboxes

In this section, we show how to construct an anonymous mailbox system within our model. As before, we assume that the users are in a particular subnet. Our goal is to provide these users (clients) with a transparent way to give e-mail addresses to outside parties (servers), which maintain the properties of the aliases (anonymity, protection from dossiers, etc). For example, a client might want to register at a (Web-site) server for mailing-lists, personalized news, etc. Such an e-mail address provides a server with the means to initiate interaction with a client by sending an e-mail message to the client.

We first consider a setting with the Janus engine on the gateway (Figure 1). We propose that the Janus engine computes "$a^u_{i,j}$@subnet-domain" as $c_i$'s e-mail address to be used with $s_j$. We further suggest storing a mailbox for each such active $(c_i, s_j)$ pair on the subnet's gateway, such that an owner of a mailbox is only identified by the respective alias. Messages are stored in these mailboxes, passively awaiting clients to access them for retrieval. We require that (1) given a previous, client-initiated interaction, a server can send data to the mailbox created for the (client, server) pair, (2) the Janus engine (upon being presented with $(c_i, p_i)$) lets a client $c_i$ retrieve the messages in *all* of her mailboxes without remembering a corresponding list of servers, (3) neither the Janus engine nor the mailboxes compromise on the property that the server must not store sensitive data (see Section 2). In particular, the knowledge of e-mail headers of messages (which contain $a^u_{i,j}$ and $s_j$) does not reveal client identity $c_i$. We show that the Janus function can be used to overcome the apparent contradiction of requirements (2) and (3). Note that the secrecy of the actual data stored *within* a mailbox is an orthogonal issue and can be solved, for example, by using PGP. For the setting of a Janus engine on each client (Figure 2), most of the scheme above remains unchanged with one important exception: When a client wants to retrieve her messages, the local Janus engine instructs the gateway, which mailboxes to access and hence $p_i$ is never revealed to the gateway.

**Data Structures for $(c_i, s_j)$-mailbox:** Let $a^m_{i,n_i} = \pi_M(\mathcal{J}(c_i, n_i, p_i, m))$, where we use the tag $m$ for the "mail index", $n_i$ an integer indexing $c_i$'s mailboxes, and $\pi_M$ a corresponding injective function to map the output of $\mathcal{J}$ into a suitable range. We explain the extensions in turn below. The following record $R$ is stored with the $(c_i, s_j)$-mailbox. $R$ has three fields: (1) $R_{alias} = a^u_{i,j}$, (2) $R_{index} = a^m_{i,n_i}$, (3) $R_s = s_j$. The argument $n_i$ in (2) indicates the index of the mailbox created for client $(c_i, p_i)$ and server $s_j$. The record $R$ (and consequently the mailbox) can be accessed both via $R_{alias}$ or $R_{index}$. The $R_{alias}$ field contains the name of the mailbox that is used for messages sent from $s_j$ to the client $c_i$. A second data structure, stored together with the mailboxes, holds a counter $C_i$ for each of the clients $(c_i, p_i)$. $C_i$ is the number of mailboxes the client $(c_i, p_i)$ has established so far. These counters are initialized to 0. Note that $0 < n_i \leq C_i$. The counter itself is indexed by $a^m_{i,0}$, so that the Janus engine, upon being presented with $(c_i, p_i)$, can easily find it.

**Creating a Mailbox:** Whenever the client $c_i$ instructs the Janus engine to give out an e-mail address for $s_j$, the engine checks if a record $R$ with $R_{alias} = a^u_{i,j}$ already exists in the first data structure. If it does not exist, then the engine retrieves the counter $C_i$ by accessing the second data structure with the key $a^m_{i,0}$. If no $C_i$ is found, it is initialized to zero. The counter $C_i$ is incremented and a new record is $R$ created, with: $R_{alias} = a^u_{i,j}$, $R_{index} = a^m_{i,C_i}$, $R_s = s_j$. Afterwards, the engine stores the updated value of $C_i$ in the second data structure with key $a^m_{i,0}$. Finally, the Janus engine create a new mailbox under the name of $R_{alias}$.

**Retrieving Mail:** Whenever client $c_i$ connects to the Janus engine, it will retrieve all of $c_i$'s accumulated e-mail messages. The engine first retrieves the counter $C_i$ by accessing the second data structure with the key $a^m_{i,0}$. Then it retrieves all records $R$ with $R_{Index} = a^m_{i,\gamma}$ for $0 < \gamma \leq C_i$. For each such record $R$, Janus retrieves the corresponding mailbox and presents it, together with $R_s$, to the client $c_i$.

The above scheme constitutes a service to store mail for any client and allows a client $c_i$ to retrieve all her mail upon presenting $(c_i, p_i)$. If $c_i$ is uncorrupted and not identifiably opened with respect to server $s_j$, then adversary $E$ cannot do better than guessing the identity of the corresponding mailbox. Furthermore, given any two such mailboxes, $E$ cannot do better than guessing whether they have the same owner. This is a simple consequence of the properties of the Janus function $\mathcal{J}$.

The above system can easily be extended to allow a client to actively send e-mail to servers using the Janus engine to generate a different address depending on the server.

### 4.3 Combining our Solution with Pseudonymous Remailers

When we allow the adversary to execute more elaborate attacks (than we introduced in our model of Section 3), such as eavesdropping or traffic analysis, a client visiting several servers within a short period of time, might become vulnerable to correlation and building of dossiers (albeit not to compromise of anonymity). Also, if a client happens to reside on a small subnet, the subnet's population might not be large enough to protect her identity. In these cases, it makes sense to combine our method with anonymous remailers or routing (for Web traffic) for enhanced protection: We can view the Janus engine as a client's "front end" to a pseudonymous remailer. It computes the different nyms on a client's behalf and presents them to the remailer. It manages all the client's mailboxes and presents incoming messages to the client. It also manages a client's public/private keys for each nym. Furthermore, even the remailer closest to the client (of a possible chain) can neither infer the client's identity nor correlate different aliases. All this remailer sees (when decrypting a reply block) is the client's alias e-mail address.

## 5 Trade-Offs and Applications

In this secton we examine the trade-off between the configurations corresponding to Figure 1, which we refer to as the *gateway approach* and to Figure 2, which we refer to as the *local approach*. We then present a few concrete applications.

### 5.1 Local vs. Gateway

The basic advantage of the *local approach* is that the Janus functionality is pulled all the way to the client's machine, minimizing outside trust. Thus, the client does not have to reveal her secret passphrase to another machine (the gateway). A client also has the flexibility to choose a mailbox location outside her own subnet, minimizing the trust in the subnet (e.g., the client's ISP). There are also a number of scenarios, where the Janus functionality is required to be on the client's machine: For example, in the realm of Web browsing, the Janus engine can be integrated with the *Personal Privacy Preferences* (P3P) standard proposal to make a P3P *persona* (see [P3P]) pseudonymous: P3P enables Web sites to express privacy practices and clients to express their preferences about those practices. A P3P interaction will result in an agreement between the service and the client regarding the practices associated with a client's implicit (i.e., click stream) or explicit (i.e., client answered) data. The latter is taken from data stored in a *repository* on the client's machine, so that the client need not repeatedly enter frequently solicited information. A *persona* is the combination of a set of client preferences and P3P data. Currently, P3P does not have any mechanisms to assist clients to create pseudonymous personae. For example, a client can choose whether to reveal his/her real e-mail address, stored in the the repository. If the e-mail address is not revealed, the Web-site cannot communicate with the client and if the e-mail address is indeed revealed, the Web-site has a very good indication on the identity of the visitor. Using a Janus engine provides a new and useful middle ground: The data in repository corresponding to usernames, passwords, e-mail addresses, and possibly other fields can be replaced by macros which, by calling the Janus engine, expand to different values for different Web-sites and thus create a pseudonymous personae for the client.

For the case of the *gateway approach*, we note that the Janus engine does not have to be distributed throughout the subnet. Thus, the clients do not have to download or install any software and no maintaince is required, also giving *scalability*: when the population in the subnet grows, it enables to easily add gateway machines (helped by *Forward Secrecy* property). The proxy might also provide *alias management capabilities* in the case where the gateway is for a corporate intranet: Such capabilities might include two clients to share their aliases for all the servers, a client to transfer one or more of his/her aliases to another client, or even two clients to *selectively* share some of their aliases. For example, when going on vacation, a manager might use such functionality to have an assistant take over some of his daily correspondence. Such alias management functions have the potential to considerably simplify login account and e-mail management in big intranets. We note that to achieve this potential, state has to be added to the proxy design, which goes beyond the scope of this paper.

### 5.2 Applications

**Web browsing**. There is a growing number of web-sites that allow, or require, users to establish an account (via a username and password) before accessing the information stored on that site. This allows the web-site to maintain a user's personal preferences and profiles and to offer personalized service. The *Lucent Personalized Web Assistant* is an intermediary Web proxy, which uses a Janus engine to translate a user's information (user's e-mail and passphrase) into an alias (username, password, email) for each web-site. Moreover, this alias is also used by the web-site to send e-mail back to a user. More details of this work can be found in [GGMM97] and at `http://lpwa.com:8000/`. The intended configuration for this project is the gateway approach of Figure 1. We note that such concrete applications typically execute in conjunction with many other mechanisms. For instance, Web browsing based on the HTTP protocol interfaces, among others, with SSL for encrypting the communication and with Java and JavaScript for downloadable executables. Each such interface can potentially undermine the pseudonymity of the client-server interaction. In the case of SSL, the proxy can spoof SSL on behalf of the internal client (see [SSL-FAQ]). The proxy can initiate SSL between itself and other servers and thus maintain the client's pseudonymity. Both Java applets and JavaScript scripts, when downloaded from a server by a client, can potentially obtain compromising client information. Research is being conducted which might lead to include *customizable security policies* into these languages (see [GMPS97, AM98]). A client can then choose a policy strict enough to preserve his/her pseudonymity. Another approach is to bundle an LPWA proxy with an applet/script blocking proxy, as described, e.g., in [MRR97]. In summary, it is necessary to consider all possible interfaces, and offer encompassing solutions to clients.

**Authenticated Web-traffic.** Consider a Web site which offers repeated *authenticated* personalized stock quotes to each of its subscribers. The value of a single transaction (e.g., delivery of a web-page with a customized set of quotes) does not warrant the cost of executing a handshake and key distribution protocol. A lightweight security framework for extended relationships between clients and servers was recently proposed [MMS97]. The Janus engine provides a persistent client-side generated shared key for each server, used in application-layer primitives. Hence, no long-term secure memory is needed on the client-side, enabling scalability and mobility.

## Acknowledgments

We thank David M. Kristol for his insights and for his many contributions to the design implementation of LPWA, which uses the Janus engine. We are grateful to Russell Brand for thought-provoking discussions.

## References

[A93] R. Anderson, The classification of hash functions. *Cryptography and Coding IV*, pp. 83–94, December 1993.

[AM98] V. Anupam, A. Mayer, Security of web browser scripting languages: Vulnerabilities, attacks, and remedies. In *Proc. 7th USENIX Security Symposium*, 1998.

[Anon] The Anonymizer. `http://www.anonymizer.com`.

[B96] A. Bacard, Anonymous Remailer FAQ. `http://www.well.com/user/abacard/remail.html`.

[BKR94] M. Bellare, J. Kilian, P. Rogaway, The security of cipher block chaining. *Advances in cryptology – CRYPTO'94*, Springer Verlag LNCS 839, pp. 341–358.

[C81] D. Chaum, Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, **24**(2), 1981, pp. 84–88.

[C85] D. Chaum, Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, **28**(10), October 1985, pp. 1030–1044.

[DKR97] J. Daemen, L.R. Knudsen, V. Rijmen, The block cipher SQUARE. *Fast Software Encryption'97*, Springer-Verlag LNCS, to appear.

[E96] A. Engelfriet, Anonymity and privacy on the internet. `http://www.stack.nl/galactus/remailers/index.html`.

[GGMM97] E. Gabber, P.B. Gibbons, Y. Matias, A. Mayer, How to make personalized web browsing simple, secure, and anonymous. *Financial Cryptography'97*, Springer-Verlag LNCS 1318.

[GMPS97] L. Gong, M. Mueller, H. Prafullchandra, R. Schemers, Going beyond the sandbox: An overview of the new security architecture in the Java Development Kit 1.2. In *Proc. USENIX Symposium on Internet Technologies and Systems*, 1997.

[GT96] C. Gulcu, G. Tsudik, Mixing email with babel. In *Proc. ISOC Symposium on Network and Distributed System Security*, 1996.

[GWB97] I. Goldberg, D. Wagner, E. Brewer, Privacy-enhancing technologies for the internet. In *Proc. Compcon*, 1997.

[KGGMM98] D.M. Kristol, E. Gabber, P.B. Gibbons, Y. Matias, A. Mayer, Design and implementation of the Lucent Personalized Web Assistant (LPWA). Submitted for publication.

[LM91] X. Lai, J. Massey, Markov ciphers and differential cryptanalysis. In *Proc. EUROCRYPT'91*, Springer Verlag LNCS 437, pp. 17–38.

[MMS97] Y. Matias, A. Mayer, A. Silberschatz, Lightweight security primitives for e-commerce. In *Proc. USENIX Symposium on Internet Technologies and Systems*, 1997.

[MRR97] D. Martin, S. Rajagopalan, A. Rubin, Blocking Java applets at the firewall. In *Proc. ISOC Symposium on Network and Distributed System Security*, 1997.

[PO95] B. Preneel, P.C. van Oorschot, MDx-MAC and building fast MACs from hash functions. *Crypto'95*, Springer-Verlag LNCS 963, pp. 1–14.

[PW85] A. Pfitzmann, M. Waidner, Networks without user observability - design options. *Eurocrypt'85*, Springer-Verlag LNCS 219, pp. 245–253.

[P3P] P3P Architecture Working Group, General Overview of the P3P Architecture. `http://www.w3.org/TR/WD-P3P-arch`.

[R95] R. Rivest, The RC5 encryption algorithm. *Fast Software Encryption*, Springer Verlag LNCS 1008, pp. 86–96, 1995.

[RR98] M.K. Reiter, A.D. Rubin, Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, **1**(1), June 1998.

[S96] D. Simon, Anonymous communication and anonymous cash. *Crypto'96*, Springer Verlag LNCS 1109, pp. 61–73.

[SGR97] P. Syverson, D. Goldschlag, M. Reed, Anonymous connections and onion routing. In *Proc. IEEE Symposium on Security and Privacy*, 1997.

[SSL-FAQ] SSL-FAQ at `http://www.consensus.com/security/ssl-talk-sec03.html`.

# Secure WWW Transactions Using Standard HTTP and Java Applets

F. Bergadano
*Dipartimento di Informatica, Università di Torino, Italy*
bergadan@di.unito.it
B. Crispo
*Dipartimento di Informatica, Università di Torino, Italy*
*Cambridge University Computer Laboratory, England*
bc201@cl.cam.ac.uk
M. Eccettuato
*Dipartimento di Informatica, Università di Torino, Italy*

## Abstract

Can users access information on the Web securely with their unchanged, normal broswers, and yet without relying on the cryptographic software contained in those browsers? In this paper we show that this is possible, with a communication architecture based on Java applets. This is important, because cryptographic functions need be separated from both the user interface and the communications routines. It must be possible to acquire the source code for the relevant modules and alternative software vendors must be available, in order to avoid hidden trapdoors and undetected implementation problems. Our approach is alternative to solutions at the protocol level (e.g., SSL), because the unchanged HTTP/TCP/IP stack is maintained. Moreover, it does not require the installation of proxies.

## 1 Introduction

The explosive growth in information that becomes available through the Web has led to the development of new applications. Some of those applications, such as electronic commerce or teleworking, are particularly critical because they require secure communications between clients and servers. For those applications, WWW transactions must offer security services such as authentication, secrecy, data integrity and non repudiation. Some browser vendors and standardisation groups have already proposed some solutions to this problem, in a way that we consider still unsatisfactory from users' point of view.
Standardisation groups have proposed to secure HTTP transactions at the protocol level. Even if these solutions enhance the security of the communications between WWW clients and servers, they are hidden below the application level, where users are often unable to access data and system components. Moreover, new protocols require new client and server software, that may not become as widespread as standard HTTP-based implementations. Browser vendors, sometimes, provide solutions at the application level. However, for commercial reasons they may not provide the source code of their solutions. The choice of cryptographic modules may also be limited by local regulations.
We believe that new approaches are needed.

In this paper we describe a framework based on Java applications and Java applets [3] to secure HTTP transactions. We have implemented a system that allows users to perform all the encryption and authentication work outside the browser, by using applets and other locally installed software.

Software is also installed on the WWW server and performs corresponding encryption and authentication procedures. All the software we use can be easily studied and analysed. Moreover users can eventually integrate their own modules to perform the cryptographic operations without affecting the validity of our approach. Our solution enforces strong security without requiring modifications in the existing HTTP protocol [7, 12] or in the available commercial browsers. Any browser supporting Java may be used.

## 2 Related Work

There are already several different approaches used to secure HTTP transactions. They can be classified as follows:

1. **Application protocol solutions.** The HTTP protocol is changed so that the header includes key management features and encryption/signature becomes possible. A well known example of this kind is the SHTTP protocol [11]. Another modification of HTTP that supports server group authentication is described in [17].

2. **Session layer solutions.** HTTP communication is implemented on top of a modified transport API, that guarantees end-to-end privacy and authentication. A well known example following these principles is SSL [13, 10].

3. **Transport layer solutions.** HTTP communications can also be secured at the transport level, securing the underlying TCP connections using tunnelling protocols such as PPTP [15] or SSH [21] [20], or at the network level using, e.g., IPv6 [9]. The Point-to-Point Tunnelling Protocol was designed to provide authentication and encryption over a public TCP/IP network using the common Point-to-Point Protocol (PPP), thus creating a Virtual Private Network (VPN). PPTP works by encapsulating the virtual network packets inside of PPP packets, which are in turn encapsulated in Generic Routing Encapsulation packets sent over IP from the client to the gateway PPTP server and back again. PPTP does not specify particular algorithms for authentication and encryption. Instead, it provides a framework for negotiating particular algorithms. In practice most commercial products use the Microsoft Windows NT version of the protocol because it is already a part of the operating system. Even if PPTP is secure in theory, a recent flow discovered in its Microsoft implementation [20] suggests to evaluate this solution carefully. The SSH protocol can be used as a generic transport layer encryption mechanism, providing both host authentication and user authentication, together with privacy and integrity protection. SSH uses a packet-based binary protocol that works on top of any transport that will pass a stream of binary data, e.g. TCP/IP. It was originally designed to provide secure remote login but the current version also supports secure forwarding of arbitrary TCP/IP connections. The SSH server can listen for a socket on the desired port, dedicated for this purpose, automatically forwards the request and data over the secure channel, and makes the connection to the specified target port from the other side. If the target port is the port dedicated to the HTTP server, SSH can be used to secure HTTP connections. IPv6 defines two headers (Authentication and Encapsulation Security Payload) to support authentication and encryption at network level.

4. **Architectural solutions.** On the HTTP client side, the browser is configured to use an HTTP proxy. User requests will then be forced through the proxy, where encryption and signature services are available. On the HTTP server side, two solutions are possible: (1) a proxy is installed that handles encryption and signatures, and exchanges data with the actual server, or (2) the HTTP server is modified so as to guarantee secure communication with the client side.

5. **Application layer solutions.** Our proposal, uses no proxies, does not

change HTTP or socket APIs at the TCP level. Instead, HTTP communications are forced through a separate TCP connection with the help of applets within the normal browser environment. The most similar approach in the existing literature is found in [19], where CGI [14] programs are used instead of applets and secure communication is achieved with PGP [22].

Each approach has drawbacks and advantages, that may make it appropriate for different applications and different commercial contexts. Approaches number 1 and 2 are natural and technically correct. However, they typically require the implementation of a browser that supports the proposed techniques. As a consequence, strong encryption may be prevented due to regulatory considerations by the browser vendor. Moreover, there is no public access to most browsers' source code. This requires the user's trust in the cryptographic functions implemented in the browser. If, and when, the source code is delivered, the situation will obviously be improved. However, compatibility and extension software regulation will need to be addressed with care.

Solutions cited at number 3, a part for the problems of the specific solution, all present the limit to address the problem of end-to-end authentication at the transport level. In an environment where the same workstation can be shared among different users the same authenticated connection can be easily shared by a malicious user. A solution capable to provide end-to-end authentication at the application level guarantees better security.

Our approach (number 5) and approach number 4 may use commercial browsers and HTTP servers as they are available now. Therefore, they do not suffer from the above problems. However, they introduce more overhead in the communication structure. Our approach, though, is only active when needed, i.e. web pages that do not require privacy or authentication are retrieved normally. Proxies, on the other hand, seem to be hard to enable on a case by case basis. Moreover, if proxies are implemented on a separate computer, the WWW transaction's security may be at risk on the path from the browser to the proxy.

## 3 Our Proposal

WWW transactions are usually performed by two parties: WWW browsers (clients) and httpd processes (servers). Information flows initially from a client to a server. Users click on a URL, or they fill a form and submit the data, and a request is sent to the correspondent server. Then the server processes the request and sends back to the client the information requested, usually as an HTML page, which can embed several different document formats (e.g. postscript, jpeg, mpeg, text, etc).

Our solution splits these two communications in several steps, in such a way that additional security services can be implemented. The user interface is unchanged, any Java-enabled browser will be acceptable. It is also worth to remark the differences between applets and applications in our solution. While applets have to be implemented using Java, the applications could be coded using any programming language. We have chosen to use Java also for the applications only to have an homogeneous developing environment.

In the paper we assume that we can rely upon an already existing authentication infrastructure. Thus we will not describe details related to public key distribution and management. We suppose as well that users can easily and securely get the server's public keys. We also assume that users can perform encryption/decryption operations as well as signing/verification of data.

Each step in the secure communication procedure is explained below.

- *User request (Figure 1).* Access to the remote information will be obtained by the user with the browser in the usual way: by clicking on the relevant hyper-

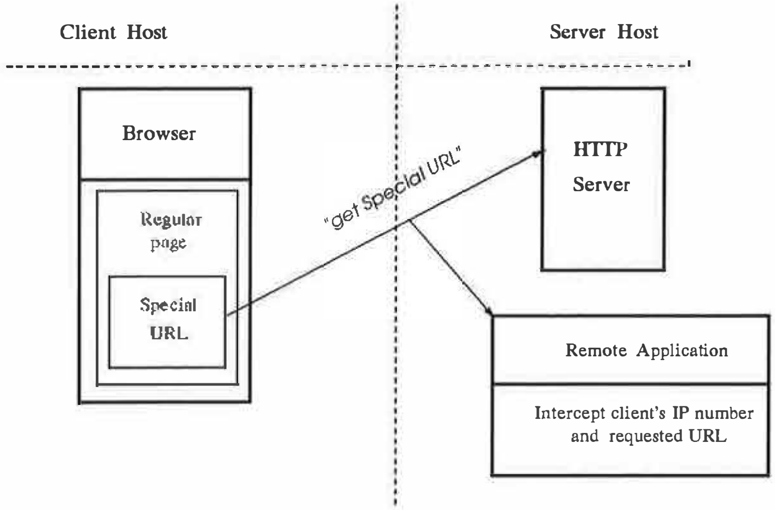


Figure 1:

link or by opening the URL directly[1]. The client's IP address, as well as the requested URL will also be intercepted, on the server side, by a running process that will later perform the actual communication, and which we shall call the *remote application*.

- *Retrieving a remote applet (Figure 2).* The selected URL will not correspond to the actual information on the server. Instead, it will reference a simple applet that the browser will run to initialise the subsequent communication environment. This will be called the *remote applet.*

  Such applets are programmed in Java and are run by the browser on the client side. In general, there has been concern regarding the security of local execution of mobile code, such as the applets. Most browsers have very strong restrictions on the execution of applets. In particular, remote applets cannot read or write local files, they cannot execute other programs in the local environment and they can only communicate with the computer they originate from. In our proposed architecture, the remote applet could be screened for security problems with even more care. The remote applet, in fact, is an extremely simple and short piece of software (see the Appendix). Moreover, the applet's code is known and is always the same, for our proposed architecture. The browser could then check whether the remote applet corresponds to a predefined pattern, and proceed to execution only if that is the case. In other words, even with very strict security policies, and in environments where running general remote applets is disallowed, the browser could agree to receive and run a predefined and limited set of specific applets. Our remote applet (received as described in Fig. 2) would be among those. At the implementation level, the browser needs to store a hash value for the allowed applets. When a remote applet is retrieved, it is run locally only if its hash value is among those previously authorised.

This whole procedure, and also the retrieval of the remote applet, is needed only if privacy and/or authentication of HTTP transactions are required. Otherwise, the base information can be immediately transmitted to the browser on the client side, and displayed to the user. The WWW server administrators will then decide whether security services are required for each of the available pages.

[1] At that point, the user will expect some kind of response from the server, e.g. some HTML page, and we shall call this the *base information*. At the level of the browser's interface, the user will obtain the base information in a transparent way, as if the information was actually stored under the selected URL. The base information will be displayed normally in the browser window when the whole procedure is completed.

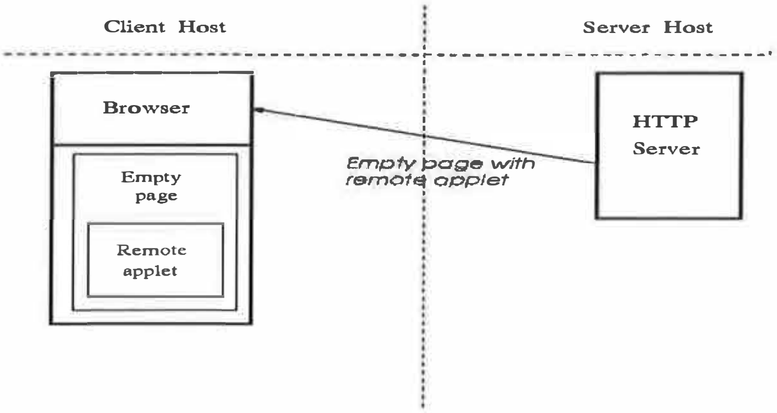


Figure 2:

If not, a page is stored under its address as advertised on the Web. Otherwise, the standard remote applet is stored in a page under that address, and the actual information is obtained by the user through the procedure explained below.

- *Opening a local applet.* The remote applet basically only does one thing: it forces the browser to open a fixed-name local URL. For instance, this URL could be "file:/security/applets/local.html". The remote applet can do this through a standard Java method called Showdocument. After this, it terminates. The remote applet is then only needed to open a local page. However, it cannot be eliminated from the architecture. First, the user will reference the base information by clicking on a remote URL, as found in hyperlinks and search index results. Therefore, the local URL, that is needed later, cannot be opened by the user in a natural way, and this must be done by the remote applet. Second, by requesting the remote URL, the browser will allow the server to record on its files the client's IP address and the URL that was referenced. This information is needed later on the server side. At this point, the remote applet has completed its life cycle and control is passed to the local URL. The local URL will reference a more substantial applet, that will control the following interaction with the user. This will be called the *local applet.* The local applet is considered trusted within this study. In fact, it is not loaded from the network at this time. It could be obtained separately, through a channel that allows for authentication.

- *Connection establishment (Figure 3).* On the HTTP client side, a ***local application*** will also be running and will communicate both with the local applet and with a remote application, running on the server machine. Such applications are actually general processes that run on the client and on the server machine. In our prototype implementation, such processes are being written as Java applications. For both of these communications, the local application will act as a server (although it runs on the HTTP client side). As a consequence, it will not need to know the IP address of the HTTP server machine. This is important as, e.g. in Netscape Communicator it is impossible to obtain this IP address without direct user input, because remote applets are very limited in their interaction with the computer they run on. In particular, current configurations would not allow the remote and local applet to communicate. This is a correct design choice in general, but we need a way around the problem of passing the HTTP server's IP address to the local application. This is accomplished by making the remote application start the TCP connection to the local application.

  On the HTTP server side, the remote application will be running and it will have obtained the IP address of the HTTP client machine as it was saved on the

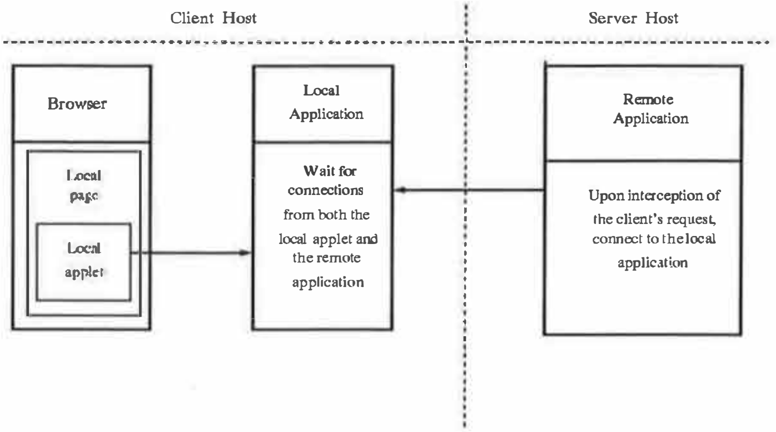

Client Host
Server Host
Browser
Local page
Local applet
Local Application
Wait for connections from both the local applet and the remote application
Remote Application
Upon interception of the client's request, connect to the local application

Figure 3:

server files as indicated in Fig. 1. Then, this application can behave as a client, and open a socket to the local application. This socket will be used for the most important part of the overall communication, including transmission of encrypted and authenticated user input and base information. As a first operation, the local and remote applications will handshake and exchange public key certificates. Protocols such as X.509 [16] or its variations [6, 8, 18, 2, 4] may be used for this purpose.

- *Secure server-to-client communication (Figure 4).* The remote application, then, will have available the URL initially referenced by the user on the client side browser, because it was stored on the server files (Fig. 1). This URL is used by the remote application to find the desired base information on the files of the HTTP server computer. Access control is performed based on user name, public key, and possibly other information obtained from the local application. In fact, the local application had previously obtained user information from the local environment, where user data and keys can be stored. At the operating system level, the user who is interacting with the browser could also undergo more sophisticated authentication procedures, that may be performed by the local applet based on passwords and/or biometric identification techniques. The local applet would then inform the local application of the verified user identity, and user data are communicated to the remote application. If the user has appropriate access rights, the base information is then encrypted and signed on the HTTP server side, and is sent to the local application.

- *Displaying the base information to the user (Figure 4).* On the HTTP client side, the local application will decrypt and authenticate the base information, and pass it to the local applet for displaying in the browser's window. The user will now see the base information, as if it were retrieved directly after clicking on the relevant link. The only difference is that the user will see a local URL with a fixed name in the browser's URL box, instead of the selected remote URL. This is against a desideratum of perfect transparency, but is actually a major advantage from a security point of view. In fact, the user can tell whether the information is local, and hence processed and received in a secure way.

- *Obtaining user input (Figure 5).* After displaying the base information, if the case requires it, the local applet will ask for user input, e.g., the user's name, credit card number and the amount to be payed. This input should then passed by the applet to the local application. However, this is not trivial. If the base information is a form, the user input will be collected by the browser, and then it will be sent to the HTTP server that is

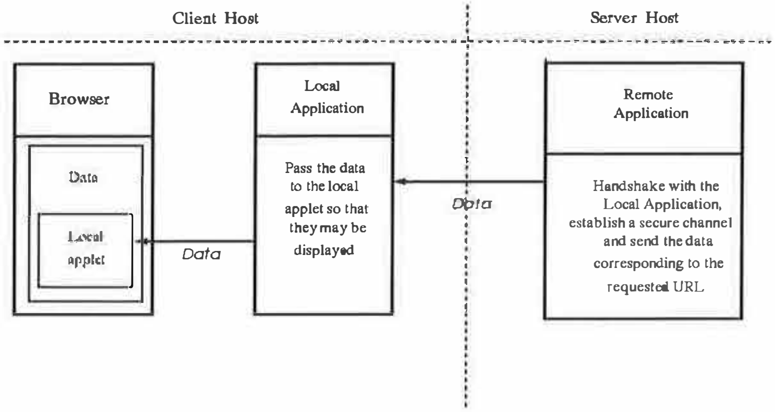

Client Host
Server Host
Browser
Data
Local applet
Data
Local Application
Pass the data to the local applet so that they may be displayed
Data
Remote Application
Handshake with the Local Application, establish a secure channel and send the data corresponding to the requested URL

Figure 4:

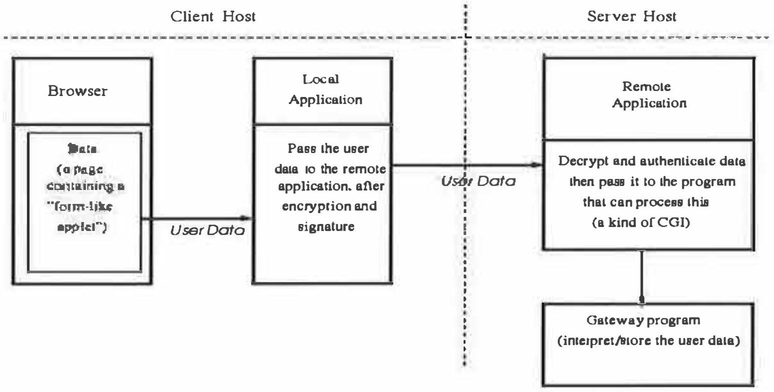

Client Host
Server Host
Browser
Data (a page containing a "form-like applet")
User Data
Local Application
Pass the user data to the remote application, after encryption and signature
User Data
Remote Application
Decrypt and authenticate data then pass it to the program that can process this (a kind of CGI)
Gateway program (interpret/store the user data)

Figure 5:

indicated in the form, in an insecure way. The local application will be cut out from this communication. We have considered two solutions to this problem: (1) installing a local HTTP server where user input could be redirected, and a general purpose CGI that transfers all data to the local application for further processing and transmission, and (2) requiring the HTTP server not to use forms, but a special parameterised Java applet that we provide and behaves like forms - when running in the browser, this applet will collect user data and immediately transfer it to the local application. Solution (2) was chosen in the implementation, because requiring an HTTP server on the original client host may be in many cases an excessive overhead. Solution (2) is cleaner and easier to install on general platforms, but requires Web sites to use our special syntax for forms.

- *Secure client to server communication (Figure 5).* The local application will now have received (1) user input obtained from our "form-like" applet and (2) the server's certified public key sent during connection establishment. The user input is then encrypted, signed, and sent to the remote application. There the plaintext information is saved in the appropriate files or passed to the needed end application, i.e. to something like a CGI script. We assume the end application does not need to send back results to the browser immediately - this can be done, but it makes the overall picture more complicated, and has not been included in the implementation. The user will then continue interacting with the browser normally, e.g. by going back to previous pages, or by clicking on new links that were displayed by means of the local applet.

One should observe that there are two distinct phases in the proposed protocol: first, an HTTP request is sent by the client and a corresponding small remote applet is received and executed, second, a local application and a remote application exchange the actual data, that may be transmitted in an authenticated and private way. The two phases are not bound in a single session, and therefore there is a spoofing opportunity here. The client may connect to one server for the first phase (no cryptography here), and the attacker could plug in her remote application for the second phase. Although this is certainly possible, we assume the client's local application will have authentication mechanisms (e.g. based on public keys and certificates) that will allow it to verify the actual identity of the remote application's owner. If the bogus application does not have the private key associated to the Web server of the first phase, as listed in the initial HTTP request, the data transfer fails. In summary, the first phase is only a trigger for the second phase, where the whole authentication procedure is performed.

## 4 Conclusion

One of the main problems that needs to be solved, before electronic commerce can become widespread and an every day habit, is to build trust among users in the safety of using the Internet. One little step in this direction is represented by securing WWW transactions. The World Wide Web is one of the most used network applications. Browsers represent user interfaces that can be used for a great variety of services.

In this paper we have presented an alternative approach to secure HTTP transactions. We believe that solutions that claim to solve security weaknesses through the use of unscrutinable and proprietary software are unsatisfactory. Many times they simply shift the line where the weaknesses can be found, from the Internet to those specific solutions [5, 1]; they often add security holes to compatibility problems. On the other hand solutions that aim to change widely used standards such as the HTTP protocol are, we think, too optimistic.

The system we have described, and that is now under testing and freely available, provides a solution without these shortcomings. The software is available at `http://maga.di.unito.it/fb/SWWWT`. A prerequisite to use our solution is for users to have a Java virtual machine running in their environment. The drawbacks of our infrastructure are: - a little degradation in performance (only when a secure channel is necessary), and - with the current version of the software, users have to agree to use a form-like applet, instead of a normal HTML form, to submit their data to the server. Servers must also adapt to this special kind of forms.

We are working to implement an alternative to this solution, using an additional, local HTTP server as mentioned in Section 3, in such a way that users can choose the solution they wish.

## 5 Acknowledgement

The authors would like to thank Ross Anderson for providing helpful comments on a draft version of this paper. The authors also thank Cosimo Anglano for helpful discussions on the implementation.

## References

[1] *Bugs in Microsoft Internet Information Server v 1.0* . `http://www.ntsecurity.com/News/bugs/iis-bug1.html`.

[2] *IETF - PKIX Working Group - Internet Public Key Infrastructure.* `http://www.ietf.org/html.charters/pkix-charter.html`.

[3] *Java Documentation and Distribution.* `http://www.javasoft.com/`.

[4] *NIST - Public Key Infrastructure Program.* `http://csrc.nist.gov/pki/`.

[5] R.J. Anderson, B. Crispo, J.H. Lee, C. Manifavas, F.A.P. Petitcolas, and V. Matyas Jr. *Global Trust Register 1998.* Nortghate Consulting Ltd., 10 Water End, Wrestlingworth, Bedfordshire SG19 2HA, England., 1998.

[6] F. Bergadano, B. Crispo, and M.T. Lomas. *Strong Authentication and Privacy with Standard Browsers. Journal of Computer Security*, 1997. Special issue on World Wide Web security - vol. 5 n. 3 pp. 191-212.

[7] T. Berners-Lee, R. Fielding, and H. Frystyk. *Hypertext Transfer Protocol – HTTP/1.0*, May 1996. RFC 1945.

[8] B. Crispo and M. Lomas. *A Certification Scheme for Electronic Commerce.* In *Security Protocol Workshop*, volume LNCS series vol. 1189. Springer-Verlag, 1997.

[9] S. Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification.* Technical report, December 1995. RFC 1883.

[10] T. Dierks and C. Allen. *The TLS Protocol, version 1.0*, November 1997. Internet Engineering Task Force Internet Draft.

[11] A. Schiffman E. Rescorla. *The Secure HyperText Transfer Protocol*, May 1996. Internet-Draft.

[12] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*, January 1997. RFC 2068.

[13] A. Freier, P. Karlton, and P. Kocher. *The SSL Protocol Version 3*, December 1995.

[14] S. Gundavaram. CGI Programming on the World Wide Web (Nutshell Handbook). O'Really & Associates, 1996.

[15] K. Hamzeh, G.S. Pall, W. Verthein, J. Taarud, and W.A. Little. *Point-to-Point Tunneling Protocol.* Technical report, `http://www.ietf.org/internet-drafts/draft-ietf-pppext-pptp-02.txt`, July 1997. Internet Draft.

[16] Information Technology - Open Systems Interconnection, Geneva. *Recommendation X.509* , June 1995. The Directory: Authentication Framework.

[17] M. Kaiserswerth, A. Hutchison, and P. Trommler. *Secure World Wide Web Access to Server Groups.* In *Proceedings of the IFIP TC6/TC11 International Conference on Communication and Multimedia Security.* Chapman and Hall, September 1996.

[18] R.L. Rivest and B. Lampson. *SDSI - A Simple Distributed Security Infrastructure.* `http://theory.lcs.mit.edu/~cis/sdsi.html`, April 1996.

[19] B. Sanderson, J.D. Weeks, and A. Cain. *CCI-based Web Security: A Design Using PGP.* In *Proceedings of the 4th International World Wide Web Conference*, December 1995.

[20] B. Schneier and P. Mudge. *Cryptanalysis of Microsoft's Point-to-Point Tunnelling Protocol (PPTP).* In *Proceedings of the 5th ACM Conference on Computer and Communication Security*, 1998.

[21] T. Ylonen, T. Kivinen, and M. Saarinen. *SSH Connection Protocol*, November 1997. Internet Draft.

[22] P.R. Zimmermann. The Official PGP User's Guide. Boston: MIT Press, 1995.

## 6 Appendix

Here is the source code of the remote applet that is downloaded by the client. This version is for Unix, but our implementation also supports Windows 95. The comments present in the actual code are omitted here. The complete code distribution, also including a demo, is available from
`http://maga.di.unito.it/fb/SWWWT`

```
RemoteApplet.java

package SecureWWW.Server;
```

```
import java.applet.Applet;
import java.applet.AppletContext;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;

/** This applet is downloaded when the
* user clicks on a reserved URL. It begins
* the whole transaction, forcing the browser
* to show the document containing the local
* applet. */

public final class RemoteApplet
  extends Applet implements SecureInterface {

 private final static String
   defUNIXlocalloaderURLname =
   "file:/usr/local/Localloader.html";

 private static URL localloaderURL = null;

 private static boolean debug = false;

 public void init() {
   debug = getParameter("debug") != null ?
    getParameter("debug").equals("true") :
    false;

   boolean noDefault =
    getParameter
    ("Localloader.NoDefault") != null ?
    getParameter
    ("Localloader.NoDefault").equals("true")
    : false;

   if ((localloaderURLName =
    getParameter
     ("Localloader.UNIX")) != null)
    try {
      localloaderURL =
        new URL(localloaderURLName);
    }
    catch (MalformedURLException except) {
     if (debug) {
      System.err.println
       ("[RemoteApplet <" + this + ">]"
       + warnSpace + "Incorrect "
       + "URL name <"

          + localloaderURLName + ">"
          + warnSpace + "for "
             + "Localloader in parameter"
          + (noDefault ? "." :": trying "
          + "default."));
         showStatus
    ("[RemoteApplet] Incorrect parameter.");
        }
```

```
    }
    if (localloaderURL == null && !noDefault)
      localloaderURLName =
        defUNIXlocalloaderURLname;
 }

 if (localloaderURL == null)

  if (!noDefault)

    try {
      localloaderURL =
        new URL(localloaderURLName);
    }
    catch (MalformedURLException except) {
    }
  else {
    System.err.println
      ("[RemoteApplet <" + this + ">]"
      + "Incorrect "
      + "parameter or no parameter,
         and no default allowed"
      + errSpace
      + "for Localloader: halting.");
    showStatus("[RemoteApplet] No valid
       URL name for Localloader.");
  }
  if (localloaderURL != null)
      if (debug) {
        System.out.println
          ("[RemoteApplet <"
   + this + ">]"
   + okSpace + "Local file URL <"
   + localloaderURL + "> created.");
        showStatus("[RemoteApplet]
          Localloader URL created.");
      }
}

public void start() {
  if (localloaderURL != null) {
     getAppletContext().showDocument
 (localloaderURL);
     if (debug) {
        System.out.println
          ("[RemoteApplet <" + this + ">]"
          + okSpace + "Local file URL <"
          + localTloaderURL + "> shown.");
        showStatus
     ("[RemoteApplet]
       Localloader URL shown.");
     }
  }
}

public String[][]
 getParameterInfo() {
 String[][] info = {
    {"debug", "boolean",
    "Controls debugging
      informations generation."},
    {"Localloader.NoDefault", "boolean",
      "To impose an exclusive value."},
    {"Localloader.UNIX","local (file) URL",
      "For Solaris, Linux..."},
    return info;
 }
}
```

# SWAPEROO: A Simple Wallet Architecture for Payments, Exchanges, Refunds, and Other Operations

Neil Daswani, Dan Boneh, Hector Garcia-Molina, Steven Ketchpel, Andreas Paepcke
*Stanford University*
*Computer Science Department*
*Stanford, CA 94305*
{daswani, dabo, hector, ketchpel, paepcke}@cs.stanford.edu

## Abstract

Most existing digital wallet implementations support a single or a limited set of proprietary financial instruments and protocols for electronic commerce transactions, preventing a user from having one consolidated digital wallet to manage all of his or her financial instruments. Commercial efforts to implement extensible digital wallets that are capable of inter-operating with multiple instruments and protocols are a step in the right direction, but these wallets have other limitations. In this paper, we propose a new digital wallet architecture that is extensible (can support multiple existing and newly developed instruments and protocols), symmetric (has common instrument management and protocol management interfaces across end-user, vendor, and bank applications), non-web-centric (can be implemented in non-web environments), and client-driven (the user initiates all operations, including wallet invocation).

## 1.0 Introduction

A number of electronic commerce applications allow end-users to purchase goods and services using digital wallets. Once a user decides to make an online purchase, a digital wallet should guide the user through the transaction by helping him or her choose a payment instrument that is acceptable to both the user and the vendor, and then hide the complexity of how the payment is executed. A number of wallet designs have recently been proposed, but we will argue they are typically targeted for particular financial instruments and operating environments. In this paper, we describe a wallet architecture that generalizes the functionality of existing wallets, and provides simple and crisp interfaces for each of its components.

In particular, the architecture we propose here has the following features. Existing proposals have some of these features, but we believe that none provides *all* of them in a comprehensive way.

*Extensible.* A wallet should be able to accommodate all of the user's different payment instruments, and interoperate with multiple payment protocols. For example, a digital wallet should be able to "hold" a user's credit cards and digital coins, and be able to make payments with either of them, perhaps using SET [1] in the case of the credit card, and by using a digital coin payment protocol in the latter case. As banks and vendors develop new financial instruments, a digital wallet should be capable of holding new financial instruments and making payments with these instruments. For instance, vendors should be able to develop electronic coupons that offer discounts on products without requiring that users install a new wallet to hold these coupons and make payments with them. Similarly, airlines should be able to develop frequent-flyer-mile instruments so that users may pay for airline tickets with them.

Many existing commercial digital wallet implementations are not extensible. They support limited, fixed sets of payment instruments and protocols, or require extra coding effort to support each instrument and protocol combination. In this scenario, end-users may need to use different wallets depending upon the payment instrument they want to use, and may even need to use different wallets to make purchases from different vendors. The CyberCash wallet, for example, only supports payments using certain credit cards and "CyberCash Coins." [2] Similarly, DigiCash's ecash Purse only supports ecash issued by a set of issuer banks [3]. The Millicent wallet only supports scrip used to make micro-payments [4]. Furthermore, while most existing wallets support at least one protocol for issuing payments, few support protocols for other types of financial transactions such as refunds or exchanges.

There do exist efforts to build digital wallets that support multiple financial instruments and payment protocols such as the Java Wallet [5] and the Microsoft Wallet [6]. In addition, some of these efforts are beginning to gain support as evidenced by initiatives such as CyberCash's development of a CyberCoin Client Payment Component (CPC) for the Microsoft Wallet, allowing users to make payments with CyberCash coins using Microsoft's Wallet. Unfortunately, these wallet architectures do not provide all of the features we describe next.

*Symmetric.* Vendors and banks run software analogous to wallets, which manages their end of the financial operations. Since the functionality is so similar, it makes sense to re-use, whenever possible, the same infrastructure and interfaces within end-user, vendor, and bank wallets. For example, the component that manages financial instruments (recording account balances, authorized uses, etc.) can be shared across these different participants in the financial operations. If the wallet components that are re-used are extensible, then we automatically get extensibility at the bank or vendor. So, for instance, an extensible instrument manager will allow the bank or vendor to easily use new instruments as they become available.

Current wallet implementations are often not symmetric. For instance, the components that make up the Microsoft Wallet are client-side objects. Seemingly little infrastructure is shared between the server-side CGI-scripts that process electronic commerce transactions, and the client-side Active/X controls that make up the wallet.

*Non-web-centric.* Interfaces should be similar regardless of what type of device or computer that the user, bank, or vendor application is running on. A digital wallet running on an "alternative" device, such as a personal digital assistant (PDA) or a smart card, for example, has substantial functionality in common with a digital wallet built as an extension to a web browser. Thus, a digital wallet in these environments should re-use the same instrument and protocol management interfaces.

Many existing wallet architectures such as the Microsoft Wallet and the Java Wallet are heavily web-centric (as they are implemented as Active/X controls or plug-ins, respectively). With the exception of the JECF's (Java Electronic Commerce Framework's) [5] recent inclusion of a smart card API, these wallets do not even begin to address issues surrounding digital wallets running on "alternative" devices (such as PDAs), or in non-web environments.

*Client-Driven.* The interaction between the wallet and the vendor, we believe, should be driven by the client (i.e., the customer). Vendors should not be capable of invoking the client's digital wallet to do anything that the end-user may resent or consider an annoyance. For example, a vendor should not be able to automatically launch a client's digital wallet application every time the user visits a web page that offers the opportunity to buy a product. Imagine what life would be like if, simply by walking into someone's store, the store owner had the right to reach into your pocket, pull out your wallet, hold it in front of you, and ask you if you wanted to buy something! A client-driven approach for building a digital wallet is important because software which customers consider "intrusive" will hinder the success of electronic commerce for all participants involved.

Some commercial wallets are not purely client-driven, since some of them can allow vendors to invoke a user's wallet simply by either: 1) having the user visit the vendor's web site, or 2) having the vendor send the user email. (See Section 6 for details.) When a vendor invokes the Java Wallet, for example, the splash page screen of the wallet applet is brought up and the user is prompted to enter her wallet password.

The wallet architecture we propose here has the features we have described. Specifically, 1) it can inter-operate with multiple existing and newly developed instruments and protocols; 2) it defines standard APIs (Application Programming Interfaces) that can be used across commerce applications for instrument and protocol management; 3) it builds a foundation general enough to implement digital wallets on "alternative" devices in addition to wallets as extensions to web browsers; and 4) it ensures that electronic commerce operations, including wallet invocation, are initiated by the client. Our contribution is not a set of "new" services for wallets, but rather a flexible architecture that incorporates the best of existing ideas in a clean and extensible way. To verify some of our functionality claims, we have implemented this architecture in Java and C++. The Java version supports most of the features described in the body of the paper. In addition, while the C++ version implements only a subset of the features described here, it does provide support for digital wallets to run on non-traditional devices such as PDAs [17]. These implementations run on the Windows and PalmOS platforms, and implementation details are described in the body of the paper.

## 2.0 Terminology

In this section, we briefly define some terminology necessary for understanding our wallet architecture. The wallet architecture is described in the next section.

## 2.1 Instrument Instance & Instrument Class

An *instrument instance* (or, *instrument*, for simplicity) is a collection of state information representing economic value that a protocol can operate on as part of an electronic commerce transaction. For example, "Gary's Citibank Mastercard" is an instrument whose state is made up of his full name, credit card number, and expiration date. The instrument may also store other information such as his billing address.

It is important to note that an instrument, in the context of this paper, is a digital proxy for an instrument in the "physical" world. "Gary's Citibank Mastercard" is, in reality, an agreement or contract between Gary and Citibank which may be made up of a signed credit card application in addition to other documents such as a contract stating Gary's credit limit and the terms of the agreement. The digital representation of this instrument, however, only contains state parameters that are relevant for conducting commerce transactions with that instrument, and the instrument's digital representation need not contain the actual contract. In the case of "Gary's Citibank Mastercard", for example, the digital instrument may only contain those state parameters necessary for the SET protocol [1] to execute an online payment operation.

Each instrument belongs to an *instrument class*. An instrument class defines the structure of the state information necessary to store an instance of a given instrument, as well as behavior that is common to all instruments of that class. Examples of instrument classes are CyberCoin [2], ecash [3], or Mastercard. "Gary's Citibank Mastercard" is an instance of the Mastercard instrument class.

## 2.2 Protocol

A *protocol* defines a sequence of *steps* that accomplish a particular *operation* using a specified instrument. In each step, the protocol may send information to a peer, or process information locally. For example, in one step the protocol may create a certificate containing the user's account number and payment amount. In the next step, it may send the certificate to the peer.

The work of a protocol is to execute a correct sequence of steps to accomplish a requested operation; the sequence is not necessarily static and pre-determined, but may vary dynamically depending upon requests and responses sent between the parties executing the protocol. In general, a payment protocol is one that supports a PAY operation and whose sequence of steps results in a transfer of economic value between two or more parties. SET is a payment protocol that may be used to transfer monetary value from a bank to a vendor's account, while concurrently (and atomically) debiting the user's credit card account, under the condition that the resulting balance does not exceed the user's credit limit.

A protocol may be defined to be compatible with one or more instrument classes. The SET protocol is compatible with both credit card and debit card instrument classes, and may be used to execute payment operations with both credit cards and debit cards (see Figure 1). The CyberCash Protocol, on the other hand, may only be compatible with the CyberCoin instrument class.

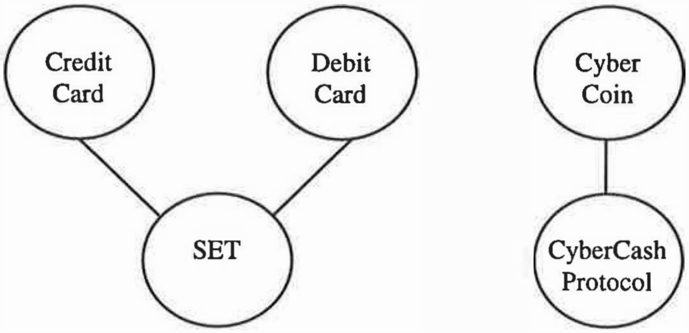


*Figure 1: Protocol & Instrument Class Compatibility*

## 2.3 Client

A *client* is a human user or a software agent. A software agent may allow the user to make online purchases or participate in online auctions, for instance.

## 2.4 Digital Wallet

A *digital wallet* is a software component that provides a client with instrument management and protocol management services. Instrument management and protocol management are defined in Section 3, but, in brief, are services that allow the wallet to 1) install and uninstall instrument classes and protocols, 2) create, update, and delete instruments and protocols, and 3) execute protocols. Digital wallets are capable of executing an operation using an instrument according to a protocol. A digital wallet presents its client with a standard interface of functions; in the case that the client is a human user, this standard interface of functions may be accessed through a graphical user interface (GUI).

A digital wallet is linked into an end-user, bank, or vendor application and provides the application with instrument management and protocol management services. The digital wallets that are linked into vendor and bank applications provide these management services in the same way that end-user digital wallets do. A vendor's digital wallet, however, may be part of a much larger software application that is integrated with order and fulfillment systems. Similarly, a bank's digital wallet may be part of a larger application that is integrated with general ledger, profit & loss, and reconciliation systems.

Furthermore, a wallet is not limited to being a plug-in or applet or some other extension of a web browser. A digital wallet with a graphical user interface may also run as an application on its own. A digital wallet may also run on computers that are not connected to the Internet such as smart cards or personal digital assistants. The user interface to the digital wallet may vary in such cases. In the case of a PDA, for example, the digital wallet may have a pen-based user interface. In the case of a smart card, the digital wallet may have no user interface at all. Nevertheless, in each case, the set of functions that the digital wallet's interface presents to its client should be the same.

## 2.5 Peer

A *peer* is a remote entity that may be an end-user, vendor, or bank wallet that is capable of performing operations on instruments according to a protocol.

## 2.6 Session

A *session* is an interaction between two peers, and state information that may be built up over time as a result of interaction between the two peers may be stored in a `Session` object. One peer is said to initiate a session, while the receiving peer is said to service the session. The `Session` object keeps track of which peer is the initiator and which is the servicer.

## 3.0 SWAPEROO Architecture

In the following, we present an *extensible*, *symmetric*, *non-web-centric*, and *client-driven* architecture for digital wallets.

In the SWAPEROO architecture, the interaction between a client wallet and a peer wallet roughly works as follows: Once a session is initiated by the client and the peer wallet prepares to service the client, the client can determine what instrument classes are available on the peer wallet, and then select an instrument class that is common to both peers. After an instrument class is selected, protocol management functions are called to determine what available protocols may be used to conduct operations on an instrument of the selected class. Depending upon what protocols are shared, a protocol is selected. The protocol supports certain operations for the selected instrument class, and the client may invoke those operations on an instrument instance. This interaction is described in detail in Section 5.

A digital wallet is an object that has four required key architectural component objects: a Profile Manager, an Instrument Manager, a Protocol Manager, and a Wallet Controller (see Figure 2).

In Figure 2, objects within the dotted lines are the core components of the wallet object[1]. We assume that communication between the core components of the wallet object is secure such that sensitive data structures containing private information about users and their instruments may be passed between objects within the wallet. In a real wallet implementation, this "boundary" around the secure components of a wallet may be supplied by the operating system by having the core components reside within the address space of a process. This approach, of course, assumes that the operating system is trusted and safe. A trusted operating system will not itself attempt to compromise the security of the wallet, and a safe operating system allegedly has no loopholes which would allow other malicious software it is running to compromise the security of the wallet by reaching into its address space.

Since code modules that implement instruments and protocols may be obtained from different sources, they cannot all be mutually trusted. In particular, it should not be possible for instruments or protocols developed by a software vendor to compromise the privacy, security, or functionality of instruments or protocols developed by another software vendor. At the same time, it is beneficial to have instruments and protocols

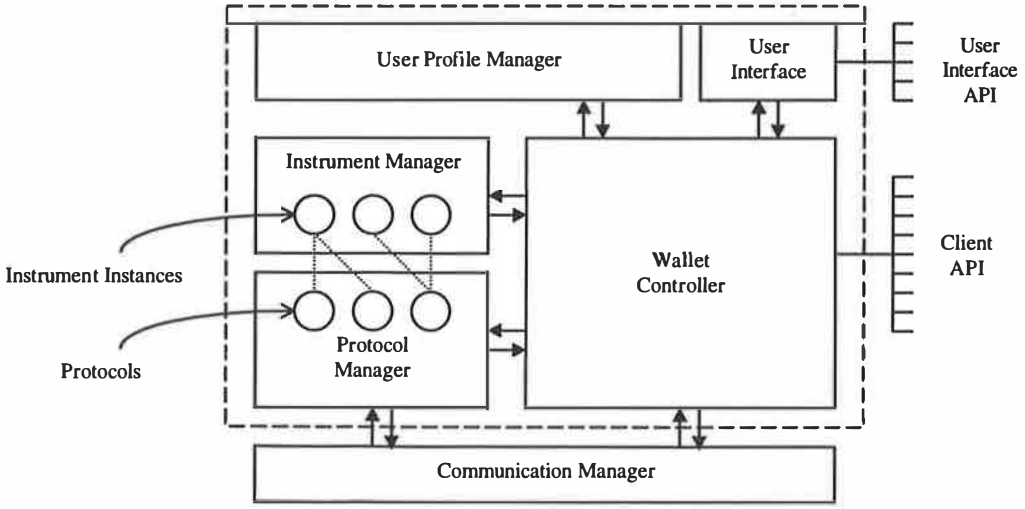


*Figure 2: The SWAP Generalized Digital Wallet Architecture*

[1] Although the User Interface is a core component when present, it is not a required component.

run within the same address space such that they can exchange private data efficiently.

Two approaches should both be taken to accomplish these conflicting goals. Code modules that implement instruments and protocols should, first of all, be digitally signed by their source (i.e. the software vendor that developed the module). As part of the installation process of such a code module into a wallet, the signature on the code module should be checked to determine if the module can be trusted. Secondly, after installation of the module, a capabilities-based security model needs to be employed to protect modules of code from each other. In such a model, a called object would be able to authenticate the object calling it by verifying that object's digital signature. The security model may also provide support to allow an object calling another object to verify the called object's digital signature. Under such a security model, objects that implement various instruments and protocols may authenticate each other at run-time to prevent potentially "malicious" calls from taking place. Implementing such a security model was beyond the scope of our work. A Java-version of such a security model is addressed by [16], and could be re-used within our work.

Objects that are outside the dotted lines reside in a different address space. However, under a capabilities-based security model in which these external objects are only granted the appropriate capabilities to access privileged data, they may optionally reside in the address space of the wallet process itself, or can be dynamically-linked into the wallet process.

All components of the wallet are briefly described below except for the wallet's Cryptographic Engine (which has been excluded from Figure 2, since all components of the architecture within the wallet may use the Cryptographic Engine to encrypt sensitive data). The Cryptographic Engine resides within the wallet's address space.

1. The *Instrument Manager* manages all of the instrument instances contained in the wallet, and may be queried to determine which instrument classes and instances are available to execute a given payment or other operation.

2. The *Protocol Manager* manages all of the protocols that the wallet may use to accomplish various operations, and invokes protocols to carry out the interaction between the digital wallet and the vendors and banks. The Protocol Manager relies on the Communication Manager to process low-level communications requests with other computers representing banks and vendors.

3. The *Wallet Controller* presents a consolidated interface for the entire wallet to the client by coordinating the series of interactions between the Profile Manager, Instrument Manager, and Protocol Manager necessary to carry out high-level requests received from the client, such as "purchase a product." The Wallet Controller hides the complexity of the other components of the wallet, and provides a high-level interface to the client. A non-human client, or software agent, can make method calls on the Wallet Controller's interface through the Client API. A human client may use a graphical user interface (GUI) which may make method calls on the Wallet Controller. The Wallet Controller also handles end-user authentication and access control for operations in the wallet.

4. The *User Profile Manager* manages information about clients and groups of clients of the wallet including their user names, passwords, ship-to and bill-to addresses, and potentially other profile information as well. In addition, the Profile Manager keeps access control information about what financial instruments each user has the authority to access, and the types of operations specific users have the privilege to execute with them.

5. The *Communication Manager* provides the wallet with an interface to send and receive messages between a wallet and a peer by setting up a "connection" with a remote Communication Manager. The Protocol Manager builds on top of the "connection" abstraction to support the concept of a session. A "connection" is typically asynchronous, while communications between peers in a session occur in (message,response) pairs where one peer sends a message, the other peer receives the message, executes some action, and returns a response. Depending upon the implementation of the Communication Manager, the messages may be sent over different types of networks using different communication protocols.

   For example, one implementation of a Communication Manager may send and receive messages over the Internet using HTTP requests and responses over a TCP/IP ethernet network. In this case, a session may be made up of a sequence of several HTTP GET messages and their corresponding responses. Another implementation of a Communication Manager may send and receive messages over an RS232 port.

Note that the Protocol Manager is responsible for making calls to the Cryptographic Engine to encrypt any data that is passed to the Communication Manager, such that the data can be securely transmitted over the communications medium. The Communication Manager cannot be responsible for encryption of sensitive data from the wallet because it is not a core component, and can be replaced by another Communication Manager to run the wallet on another device. If the Communication Manager is relied upon to encrypt sensitive data, then the Communication Manager might be replaced with a malicious Communication Manager that sends all sensitive data to an adversary.

6. The *Client API* is an interface provided by the Wallet Controller that may be used by a software agent acting on behalf of an end-user, vendor, or bank.

7. The *User Interface* provides a graphical interface to the services offered by the Wallet Controller's interface. The User Interface is an optional component of the wallet. Some devices, such as most smart cards, do not have the ability to display a graphical user interface, and hence the Wallet Controller interface must be accessed through the Client API. Note that the User Interface is a core component within the wallet because certain parts of the user interface have access to sensitive user data. For example, the edit box object into which a user enters the password to "unlock" the wallet should run within the wallet's protected address space. On the other hand, customization of the wallet's user interface presents an important branding opportunity for banks and vendors that distribute wallets.

8. The wallet's user interface exports parts of its interface as the *User Interface API* to satisfy both the privacy and customization requirements. Methods in the User Interface API may be overloaded by software vendors to render customized parts of the interface. The User Interface API also decouples the GUI so that the GUI can be run on a thin client, such as a network computer, while the core components of the wallet can be run on a server.

We will now describe each of the required core components of the digital wallet.

## 3.1 Instrument Management

The Instrument Manager is responsible for managing instrument instances, as well as information about classes of instruments. Instrument Management is made up of the following services: instrument capability determination, instrument installation, instrument storage and retrieval, and instrument negotiation. Before describing the Instrument Manager interface in detail, we will first briefly describe the instrument objects that the Instrument Manager is responsible for storing and retrieving.

An instrument may be a financial instrument that can be used to make a payment, such as a credit card, debit card, or electronic coin. More generally, however, an instrument is made up of state information representing economic value that a protocol can operate on. For example, a digital cash instrument's state can be made up of its dollar (or other currency) value digitally signed by its issuing bank. The protocol used between an end-user wallet and a vendor wallet supports a `VERIFY` operation which verifies that the cash is authentic by applying the issuing bank's public key to the coin.

While end-user, vendor, and bank wallets share many code modules, some specialization is appropriate. Instruments may have to be managed slightly differently in end-user, vendor, and bank wallets. To illustrate this, we consider a trivially simple digital cash instrument example; please note that in real systems such a naive digital cash scheme is not viable because of real-world security, efficiency, and performance considerations. In this "toy" digital cash example, every time that a vendor receives a digital coin signed by a client's private key, the vendor needs to keep track of that signature in addition to its dollar value in its digital cash instrument. On the other hand, the client may want to keep track of the vendor's signatures on coins she signed for purposes of non-repudiation in her digital cash instrument. Although this is a simple case, we can begin to see that the state information for the digital cash instrument may differ depending upon whether or not the digital cash is being stored in the end-user's wallet or in the vendor's wallet.

Consider a digital cash instrument whose instrument class is `Digital-Cash-Instrument`. We derive two subclasses from our `Digital-Cash-Instrument` to manage the different implementations of the digital cash instrument on the different peers. A `Vendor-Digital-Cash-Instrument` is stored on the vendor, and is able to store a list of client's digital coin signatures. A `Client-Digital-Cash-Instrument` is able to store the vendor's signature on the client-signed coins. Although the Vendor-Digital-Cash-Instrument and the Client-Digital-Cash-Instrument, for the most part, present similar interfaces since they both derive from Digital-Cash-Instrument, the subclasses present specialized interfaces to their respective callers to access client or vendor-specific instrument information. Note once again that this is a trivial example, and is provided simply to illustrate that the representation of the digital cash instrument may need to be specialized depending upon whether the peer is a user, vendor, or bank.

In addition to providing access to instrument information in the digital wallet's memory, the Instrument Manager provides interfaces to store and retrieve instruments to and from persistent storage. Note that the Instrument Manager may make calls, if necessary, to the wallet's Cryptographic Engine to encrypt instrument state information in preparation for writing this information to persistent storage, and for decrypting instrument state information when reading this information back from persistent storage.

Upon initialization, the Instrument Manager determines what instrument classes the wallet is capable of using by consulting a configuration file, dynamically determining this through introspection, or by accessing a Capabilities Management service [7]. Alternatively, the Instrument Manager can dynamically download an instrument class from a trusted third-party, and install it. The Instrument Manager may call the Cryptographic Engine to verify that the instrument class code is signed by the trusted third-party. Once the code supporting the appropriate instrument classes is loaded, instrument instances can be created by the user, but more often are loaded from encrypted files on the user's local hard disk, or even potentially from a file server on a network. Finally, the Instrument Manager supports methods to create, modify, commit changes to, and delete instrument instances under transactional semantics.

The Instrument Manager supports methods that query for available instrument classes to conduct instrument negotiation. Note that in our client-driven approach there is no way for a vendor to "Offer" instrument capabilities as there might be in [8], unless the vendor is explicitly queried.

## 3.2 Protocol Management

The Protocol Manager is responsible for managing protocol objects. Protocol Management is made up of the following services: protocol capability determination, protocol installation, and protocol negotiation.

Upon initialization, the Protocol Manager determines what protocols the wallet is capable of using. As with Instrument Managers, this information can typically be read from a configuration file, determined dynamically through introspection, or can be accessed through a Capabilities Management service [7]. Once this information is determined, a class representing each protocol is loaded into memory, and one instance of each protocol class is instantiated. The instantiation of each protocol instance can be delayed until the Protocol Manager needs to use that protocol.

A protocol is capable of performing operations with an instrument. The exact operations that the protocol can support during a particular session may depend upon the type of peer the wallet is connected to. Consider, for example, a digital coin instrument, and associated digital coin protocols. During a session in which the wallet is connected to a bank, the digital coin protocol may provide deposit and withdrawal operations. On the other hand, while connected to a vendor, the digital coin protocol may provide purchase and refund operations. Protocol objects are responsible for ensuring that such operations take place under transactional semantics.

Furthermore, the Protocol Manager is capable of conducting protocol negotiation with peers for a specified instrument class. That is, the Protocol Manager is capable of determining which protocols the local and remote peers share in common for the specified instrument class. The Protocol Manager accomplishes protocol negotiation with a peer through a Protocol Negotiation Protocol (PNP). Protocol Negotiation Protocols are subclasses of Protocol in our architecture, and are managed in the same way as other protocols are managed, with the exception that the Protocol Manager must successfully load a PNP upon initialization. At least one PNP must load successfully such that it may engage in protocol negotiations with peers. Additionally, although a wallet may load more than one PNP upon initialization, the particular PNP that will be used to conduct protocol negotiation must be fixed by both peers before a session is initiated, otherwise a PNP to decide which PNP to use becomes necessary, and so forth.

Protocols may be *compatible* with one or more instrument classes, and a protocol object is capable of determining whether or not it is compatible with an instrument class. Once an instrument class is chosen by the client, the Protocol Manager may query the protocol objects it manages to determine which ones are compatible with the chosen instrument class. If more than one protocol is compatible with the chosen instrument class, the Protocol Negotiation Protocol may pass the list of compatible protocols to the instrument class to allow the instrument class to determine which protocol is *optimal* for that instrument class.

To illustrate with an example, consider two protocols, `Clear-Text-HTTP-Post` and `SET-Protocol`, that are both compatible with a `VISA-Credit-Card` instrument. When the Protocol Manager calls the `Clear-Text-HTTP-Post` protocol's `compatible-With` method passing the `VISA-Credit-Card` instrument class as an argument, the return value will be `true`. Similarly, when the Protocol Manager calls the `SET-Protocol` protocol's `compatible-With` method passing the `VISA-Credit-Card` instrument class as an argument, the return value will also be `true`. Both protocols may not be available on both the client and the peer, but in the case that they are, the PNP may call the instrument class's `get-Preferred-Protocol` method passing both protocols as parameters. The instrument class is responsible for determining which protocol is optimal (using its own definition of optimal), and the `VISA-Credit-Card` instrument class may return `SET-Protocol` in favor of `Clear-Text-HTTP-Post` to obtain the best possible security for subsequent operations.

In summary, a protocol object is responsible for determining *compatibility* with an instrument class, the PNP is responsible for determining the *availability* of protocols between peers, and an instrument class can be called upon by a PNP for determining *optimality* amongst several compatible protocols available to both peers.

## 3.3 User Profile Management

The User Profile Manager stores information about clients and groups of clients of the wallet. The model we assume is one in which a client is a person or software agent that has authorization to use one or more financial instruments. It is important to note that most existing wallet implementations only allow for clients that are human users and not software agents. However, for the sake of discussion in this section, we will use the terms client and user interchangeably. A group is a set of clients that have access to a set of shared financial instruments, and each client within a group is authorized to conduct financial transactions with the set of shared financial instruments. When financial instruments may be shared between clients in a group, the instruments may or may not be used concurrently by two or more users in the group depending upon the type of the instrument. Although the User Profile Manager provides access to information about which clients are authorized to use which instruments, and who "owns" or may access which instruments, the Instrument Manager (described in Section 3.2) provides synchronized, concurrent access to use instruments, such that conflicting operations are prevented. For example, a wallet should ensure that digital cash owned by a group of clients does not get doubly spent because two clients attempt to concurrently make a payment using the same digital cash instrument instance.

An example of a group might be a corporate department. Each employee in the department may have access to a set of shared financial instruments, but depending upon an employee's position within the company, the employee may be authorized to only conduct transactions under a certain amount.

## 3.4 Wallet Controller

The Wallet Controller provides an interface to all of the services that the wallet may offer to external objects. The "outside world" cannot see, and does not have direct access to any of the components internal to the wallet such as the Instrument or Protocol Managers. In our implementation, the components of the wallet are all private data members of the `Wallet-Controller` class.

Once a Wallet Controller method is invoked, the Wallet Controller coordinates the steps that need to be carried out among the User Interface, Profile Manager, Instrument Manager, Protocol Manager, and Cryptographic Engine to execute a payment or other operation. Before accessing or carrying out an operation with instrument instance data, the Wallet Controller makes the appropriate calls to the User Profile Manager to ensure that the user involved has the appropriate privileges to carry out the operation.

## 4.0 Vendor and Bank Digital Wallets

The end-user wallet interacts with bank and vendor wallets to execute commerce transactions. The bank and vendor wallets have architectures symmetric to the user's digital wallet, as shown in Figure 3.

In place of a User Profile Manager, the bank wallet has an Account Profile Manager that allows the bank to manage non-financial information about the bank's clients. (Financial information about the bank's clients is stored in instruments in the bank's Instrument Manager.) Similar to the way in which the User Profile Manager maintains access control information about instrument instances, the Account Profile Manager does the same for the bank wallet. The Bank Controller queries the Account Profile Manager before accessing instrument instance data to determine whether or not the user has the appropriate authorization to conduct a transaction; in the case that a user's credit line has been overdrawn, or if the user's credit card has been cancelled, the Account Profile Manager would return an error response to the Bank Controller's query.

In the vendor wallet, the User Profile Manger is replaced with a Customer Profile Manager, which is used to store access control information about its customers in the case of non-anonymous transactions. For example, the Customer Profile Manager might store the user's age, and the Vendor Controller may query the Customer Profile Manager to determine if the user is above a certain age to purchase a product.

A key difference between the Wallet Controller running in the end-user application and the Bank and Vendor Controllers running in the bank and vendor applications, respectively, is that the end-user Wallet Controller drives the wallet interaction, and it is active, in that it generates requests and receives responses. The Bank and Vendor Controllers, on the other hand, are passive, in that they receive requests and generate responses. A Wallet Controller generates requests, and these requests are pushed down through its Communication Manager and out to the peer wallet. Peer wallets receive requests through their Communication Managers, and the requests are propagated to and are serviced by the Bank and Vendor Controllers.

The Instrument, Protocol, and Communication Managers used by the end-user, bank, and vendor are one and the same, and are re-used across the wallets.

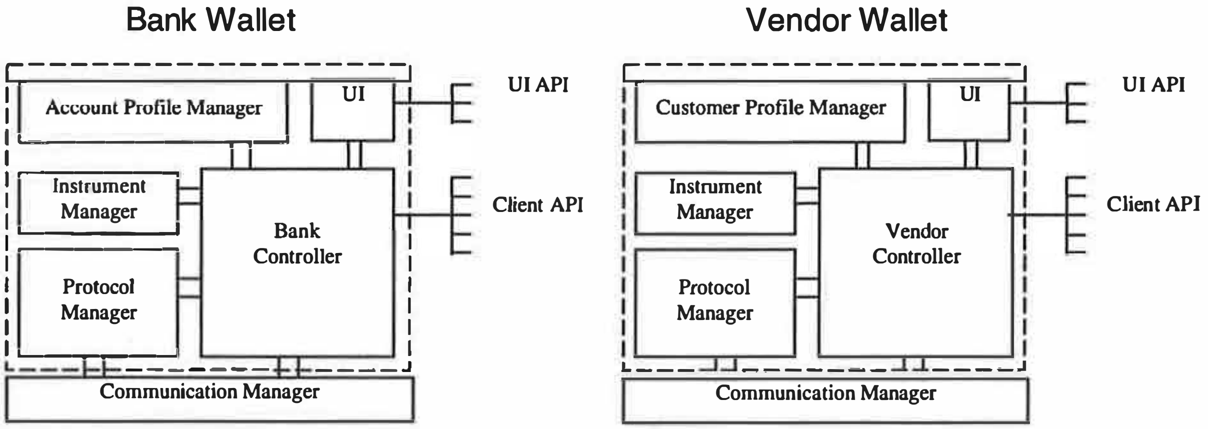


*Figure 3: Symmetric Bank & Vendor Wallet Architectures*

## 5.0 Wallet Operation & Interaction Model

In this section, we describe how our symmetric, client-driven wallets operate and interact during a session to execute a payment. (Executing any other operation such as a refund happens in a similar fashion, and may possibly require only a subset of the steps described below.) In our example here, we assume that the "ordering" phase of the shopping interaction is complete, and the necessary "invoice" (containing information about what products and/or services the end-user would like to purchase) is stored in a property list called `inv` as a set of (name, value) pairs. We will describe the process from the point of view of the end-user wallet, and we will use the diagram in Figure 4 as an aid. To the right of the states in Figure 4, we supplement the figure with the method names from our wallet implementation that an application would invoke on the Wallet Controller. Methods in boldface are public in the Wallet Controller's interface, while methods in regular font are private to the Wallet Controller. Finally, the careful reader will note that the methods seem single-threaded; it is because the wallets in our implementation are single-threaded. However, we can easily extend to the multi-threaded case by passing the Session object to each method (excluding `initiate-Session()`), or by instantiating one Wallet Controller per session.

Although we describe the interaction from the end-user wallet point of view, it is important to keep in mind that a vendor or bank's wallet can also be a client in a wallet interaction. For example, as part of a transaction with the end-user wallet, a vendor's wallet may act as a client and initiate a session with a bank's wallet to obtain credit authorization information for a purchase.

## 5.1 Initialization / Session Initiation.

When an application is launched, static initialization takes place before it starts executing. During static initialization, the wallet components including the Instrument Manager, Protocol Manager, and Profile Manager are constructed and initialized. After static initialization, the wallet may be "unlocked" by supplying login/password information and a session with a peer wallet may be initiated.

The Wallet Controller presents the user's login and password to the User Profile Manager to determine if the user should be allowed to use the wallet. The Wallet Controller also passes the user's password to the Instrument Manager so that it may use the password to decrypt instrument instances and/or the user's private key from persistent store.

To initiate a session, the `initiate-Session` method in the Wallet Controller is called, passing a `Peer` object as a parameter. (A `Peer` object contains the peer's name and details about how to set up a session with that peer.) The Wallet Controller's `initiate-Session`, in turn, calls the Protocol Manager's `initiate-Session` to initiate the session. The Protocol Manager makes calls to the Communication Manager to set up the session with the remote peer using the underlying communication mechanism.

## 5.2 Instrument Class Negotiation

The first step that takes place after session initiation is instrument class negotiation. In this step, the client's wallet can determine what instrument classes are known to both wallets by 1) determining what instrument classes are known to its Instrument Manager, 2) determining what instruments are known to the remote Instrument Manager, and 3) computing the intersection.

Those instrument classes that are available to both wallets may offer different terms and conditions for purchasing a given set of products or services. As an example, the price of a product may vary depending upon whether the customer decides to pay using a credit card or using ecash. Furthermore, extended warranties may be offered in the case that a credit card is used to make a purchase. For each available instrument class, the instrument class negotiation step also determines these terms and conditions.

To start instrument class negotiation, the application calls the Wallet Controller's `get-Instrument-Classes` method with the invoice information, `inv`, as an argument. (The invoice information is included to determine the available instrument classes because some instrument classes may not be applicable for certain types of purchases. Also, the `inv` property list is populated with the terms and conditions described above such that the terms and conditions become part of the invoice.) The Wallet Controller then calls its `get-Local-Available-Instrument-Classes` method to determine what instrument classes the local wallet supports and are available. The Wallet Controller makes this determination by, in turn, making a call to the Instrument Manager to determine what instrument classes are available to the wallet. Then, the Wallet Controller calls its `get-Remote-Available-Instrument-Classes` method to determine what instrument classes the remote peer is capable of dealing with.

The call to get-Remote-Available-Instrument-Classes results in a remote procedure call to the peer's Wallet Controller. To respond to the get-Remote-Available-Instrument-Classes procedure call, the remote Wallet Controller calls its get-Local-Available-Instrument-Classes. Other calls of the form get-Local... and get-Remote... described in the following sections work similarly. Also, in our implementation, the get-Remote-Available-Instruments call populates a "price" property stored in inv with a list of (instrument class, price) pairs to indicate the different prices that would be charged for using the corresponding instrument classes in addition to reporting the available instrument classes. The call chain for instrument class negotiation is depicted in Figure 5. Solid arrows in Figure 5 indicate method calls from the object from which the arrow originates, and dotted arrows indicates the return of control. (The Wallet Controller relies on the Communication Manager to handle the low-level details of executing the remote procedure call described above, but the actual calls that the Wallet Controller makes to the Communication Manager have been omitted from Figure 5 to keep the diagram simple. Also, arguments and return values for the methods have been omitted from Figure 5 for the same reason, but this information can be found in Figure 4.)

To digress momentarily from the end-user wallet's point of view, note that, in Figure 5, after the vendor's Wallet Controller calls its local Instrument Manager to determine what instrument classes are available, it may optionally "notify" the Vendor Application. The vendor Wallet Controller gives the Vendor Application the ability to subscribe to various events and possibly filter the results before they are returned to the end-user's Wallet Controller. Although this capability is not of crucial importance during instrument class negotiation, it is useful to notify the Vendor Application of other events, such as the successful execution of a payment. If the Vendor Application is, for example, a front-end for a vending machine, the vending machine would dispense the appropriate product after receiving notification that payment was successfully transferred.

To complete instrument class negotiation, the Wallet Controller calls get-Common-Instrument-Classes to compute the intersection of available instrument classes. The results received from get-Local-Available-Instrument-Classes and get-Remote-Available-Instrument-Classes are passed as parameters to get-Common-Instrument-Classes.

Once instrument class negotiation is completed, the application is presented with a list of instrument classes with which the user may execute a transaction. The user must select one or more instrument classes before the next step, protocol negotiation, can begin, since the choice of what protocols can be used to execute a transaction is dependent upon the instrument classes that the user selects. In a typical purchase, one instrument will be used to purchase the products and services specified in a specific inv record. However, multiple instruments, possibly of different instrument classes, may be used to make such a purchase, and protocol negotiation for each instrument class would need to be carried out.

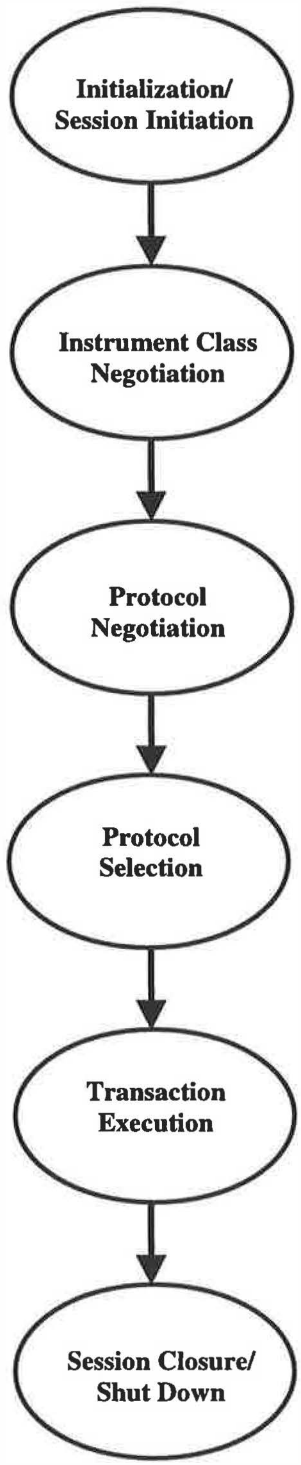
## 5.3 Protocol Negotiation

Once the instrument class has been negotiated, the application may call the Wallet Controller's negotiate-Protocol method to start protocol negotiation. The Wallet Controller's negotiate-Protocol method, in turn, calls the Protocol Manager's negotiate-Protocol method. The Protocol Manager's negotiate-Protocol method then calls the doOperation method of the currently active Protocol Negotiation Protocol (PNP), and also sends a message to the peer informing it that the local wallet is entering the protocol negotiation step. The peer will symmetrically call its PNP's do-Operation.

The default PNP that we implemented calls the Protocol Manager's get-Local-Available-Protocols method to obtain a list of protocols available that are locally compatible with the selected instrument class, and then calls get-Remote-Available-Protocols to determine the protocols that the remote wallet supports for the selected instrument class. The PNP finally calls get-Common-Protocols to compute the intersection. This default protocol negotiation mechanism is similar to instrument negotiation.

Wallet Interaction Model

Wallet Controller Methods



Initialization/ Session Initiation
Instrument Class Negotiation
Protocol Negotiation
Protocol Selection
Transaction Execution
Session Closure/ Shut Down

```
Session initiateSession(Peer aPeer)


List getInstrumentClasses (PropertyList inv)
  List getLocalAvailableInstrumentClasses(PropertyList inv)
  List getRemoteAvailableInstrumentClasses(PropertyList inv)
  List getCommonInstrumentClasses(List iclist1, List iclist2)


ProtocolRecord negotiateProtocol (InstrumentClass anInstClass)
  List getLocalAvailableProtocols(InstrumentClass anInstClass)
  List getRemoteAvailableProtocols(InstrumentClass anInstClass)
  List getCommonProtocols(List plist1,List plist2)


void setLocalTransactionProtocol(InstrumentClass instclass,
                                 Protocol protocol)
void setRemoteTransactionProtocol(InstrumentClass instclass,
                                  Protocol protocol)
List getAvailableOperations(InstrumentClass instclass,
                            Protocol protocol)


List getInstruments (InstrumentClass instclass)
void executeTransaction (PropertyList pinfo,
                         Instrument anInstrument,
                         Operation anOperation)


void closeSession (Peer aPeer)
```

*Figure 4: Wallet Interaction Model & Wallet Controller Interfaces*

Although we implemented the simple PNP above, the remote peer could respond to `get-Remote-Available-Protocols` with a list of available protocol names and the location of the signed code for those protocol classes. Such an implementation of a PNP may decide to download, dynamically link, and install a protocol that is not locally supported by the Protocol Manager if the user agrees to permit the wallet to do so. Such a PNP may expedite the case in which the intersection of locally and remotely available protocols is the null set, which would prevent the wallets from executing a transaction. The Java Electronic Commerce Framework (see Section 6), for example, employs such a mechanism as the default behavior. Finally, since the notion of a Protocol Negotiation Protocol has been abstracted out in our architecture, a PNP such as JEPI's UPP protocol (see Section 6) could be used in place of our default PNP[2].

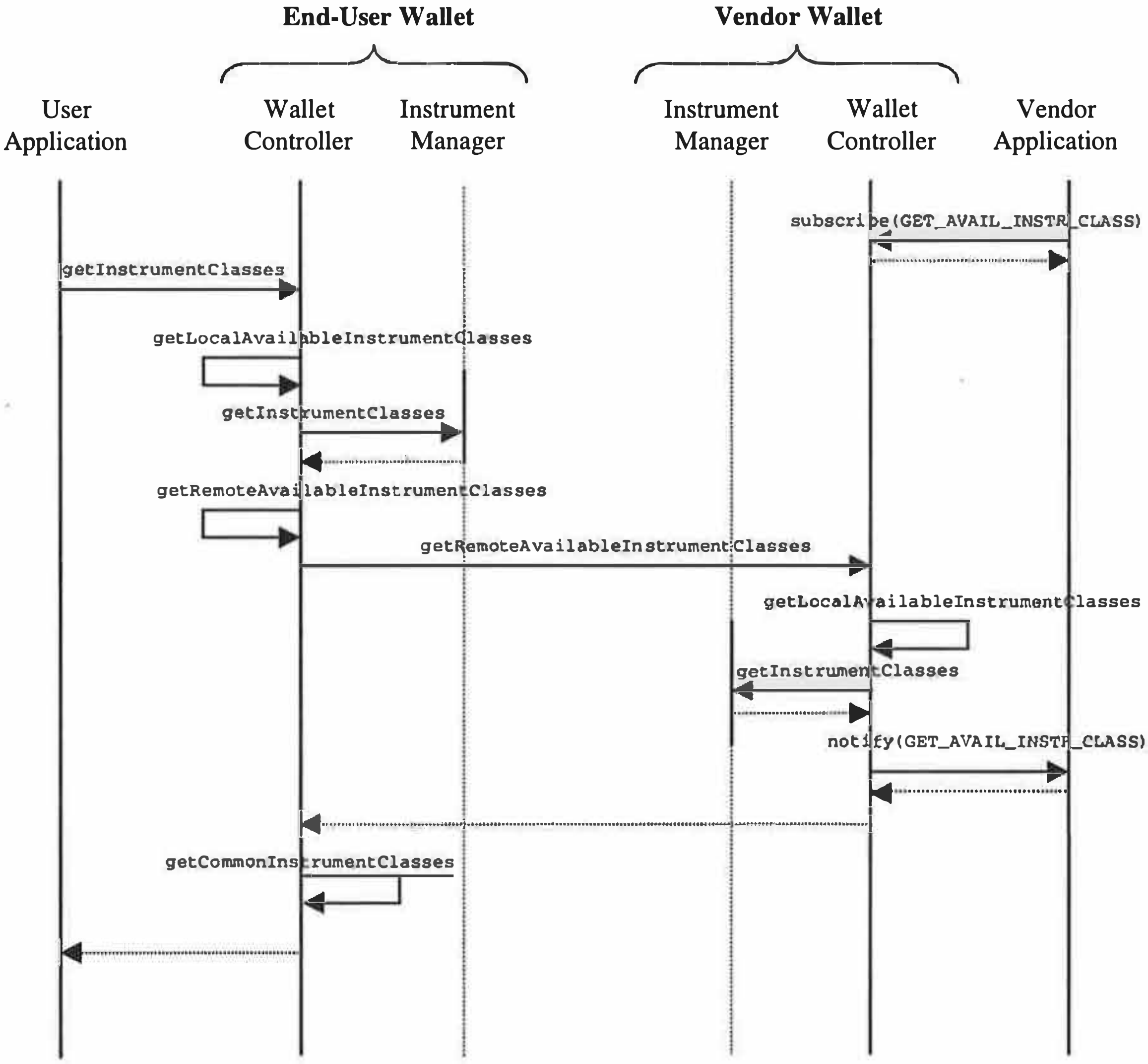

End-User Wallet
Vendor Wallet
User Application
Wallet Controller
Instrument Manager
Instrument Manager
Wallet Controller
Vendor Application
subscribe(GET_AVAIL_INSTR_CLASS)
getInstrumentClasses
getLocalAvailableInstrumentClasses
getInstrumentClasses
getRemoteAvailableInstrumentClasses
getRemoteAvailableInstrumentClasses
getLocalAvailableInstrumentClasses
getInstrumentClasses
notify(GET_AVAIL_INSTR_CLASS)
getCommonInstrumentClasses

*Figure 5: Instrument Class Negotiation Call Chain*

[2] Note that Instrument Negotiation can also be abstracted out into an Instrument Negotiation Protocol, although for our implementation we felt that demonstrating this capability with Protocol Negotiation was sufficient.

## 5.4 Protocol Selection

If the result of protocol negotiation for a given instrument class yields one specific protocol, then protocol selection is automatic, and that protocol will be used to execute transactions with instruments of the given instrument class. On the other hand, if more than one protocol is available for a given instrument class, then a preferred protocol needs to be selected by the client's instrument class.

If the PNP reports more than one protocol in common, the Wallet Controller calls the instrument class's `get-Preferred-Protocol` method as mentioned in Section 3.2. The instrument class selects an appropriate protocol among the ones available based on a variety of parameters, such as the type of device on which the wallet is running, the level of network connectivity and bandwidth available, the dollar amount of the transaction, or user preferences.

For example, when executing a transaction using a wallet on a PDA with limited processing power but with a direct connection (via a docking port or cradle) to a vendor's cash register, an unencrypted session protocol might be preferred over an encrypted session protocol since the link could be assumed to be secure, and key exchange processing overhead need not be incurred to provide for a faster transaction. Such information about device characteristics and connectivity may be obtained from a system properties table similar to the table returned by Java's `System.getProperties` method. In general, policies regarding how to carry out protocol selection based on such parameters is beyond the scope of this paper; our architecture does, however, provide a framework and the appropriate "hooks" for such policies to be implemented by implementing and/or overloading the `get-Preferred-Protocol` method.

Once the protocol is selected for the given instrument class, the Wallet Controller invokes `set-Local-Transaction-Protocol` and `set-Remote-Transaction-Protocol` with the instrument class and selected protocol as arguments. Following protocol selection, the application may inquire what operations the selected protocol makes available by calling the Wallet Controller's `get-Available-Operations` method. Protocols offer different sets of operations depending upon the instrument class over which the protocol executes and, as described in Section 3.2, whether the peer wallet is a bank or vendor.

## 5.5 Transaction Execution

At this point, the application can present the user a list of the available operations and the user can select one of them for execution. After that, the user must select an instrument instance of a previously selected instrument class on which to execute the operation. To obtain a list of possible instrument instances that the user may choose from, the application calls the Wallet Controller's `get-Instruments` method passing the previously selected instrument class as a parameter. The application presents a list of the names of each of the returned instrument instances to the user, and the user can choose one of the instrument instances.

Once an operation and instrument instance are selected, the application calls the Wallet Controller's `execute-Transaction` method with these parameters, along with the invoice information stored in `inv`. After the Wallet Controller verifies that the user has the appropriate privileges to execute the transaction by querying the User Profile Manager, the Wallet Controller calls the Protocol Manager's `execute-Transaction` method. The Protocol Manager then calls the `do-Operation` method of the appropriate protocol object. The peer wallet is sent a message informing it of the name of the protocol and operation that is to be executed, and it then starts executing that protocol. The peer will symmetrically call the appropriate protocol object's `do-Operation` method on its side. The protocol objects then execute all of the necessary actions to accomplish the operation.

The operation may or may not execute successfully. An exception will be thrown by the `do-Operation` method if the operation fails. The remote vendor or bank application will typically subscribe to its wallet's `EXECUTE_TRANSACTION` events, and will be notified of a failed operation.

Until this point in the discussion, we have described the various wallet states in succession, but it is important to note that the application may decide to negotiate over a different instrument class or select a different protocol. For instance, after transaction execution, the application may return to the instrument negotiation step to select another instrument class for the next operation. As a second example, after selecting an instrument class, and after protocol negotiation and selection takes place, the application may decide that the range of operations available from the vendor for that instrument class and protocol combination is not acceptable. At that point, the application may decide to choose another protocol instead, and protocol negotiation will be conducted once again until a protocol is chosen and another set of operations can be presented to the user. In general, after instrument class negotiation the application can return to any previous step in the wallet interaction. (The extra arrows entailed by this have been left out of Figure 4 to keep the diagram uncluttered.)

This capability can also allow payments to be executed in multiple steps with different instruments. For example, if the price of making a given purchase is constant regardless of the instrument class, the vendor may accept receiving large payments as a combination of several smaller payments of different instruments. After instrument class negotiation takes place, the end-user selects multiple instrument classes, and protocol negotiation and selection is completed for each one. The end-user then specifies the amount to be paid with each instrument, and the transaction execution step occurs for each instrument. If the payment fails for any one of the instruments, the user may be prompted to select an alternate instrument, or `REFUND` operations need to be executed for all the other instruments to abort the payment. The wallets involved must take advantage of transaction management services to ensure that recovery and rollback are executed correctly in the face of machine or network failures.

Finally, as presented above, when the user chooses an instrument class, protocol negotiation and selection are done for that instrument class, and the user then selects an instrument instance and an operation to execute. If there are many different possible instrument class, protocol, and available operation options, this approach will best guide the user through the interaction. On the other hand, if there are relatively few instrument classes, protocols, and available operation options, then forcing the user to go through all those steps may be overkill. To avoid these extra steps, the application can "hide" some steps from the user even though both wallets must go through each of the steps.

Consider an example in which the user has only a few instrument instances, and wants to quickly make a payment. After initiating a session, the user application may call `get-Instrument-Classes` and then call `get-Instruments` for each instrument class returned to obtain all instrument instances, saving the user of having to select an instrument class. The user application can present the user with a choice of all instruments. After the user chooses an instrument with which to make the payment, the application determines the instrument class from the instrument object by calling the Instrument's `get-Instrument-Class` method, and then does protocol negotiation and selection for the instrument class. If it turns out that no protocol is available for that instrument class, an error is reported to the user. Similarly, if it turns out that a protocol can be selected, but that the `PAY` operation is not available, an error is reported to the user. If a suitable protocol can be selected, and the `PAY` operation is available, the user is saved from having to explicitly select an instrument class.

## 5.6 Close Session / Shut Down

In general, the application may close the session at any time. In the typical case, the application may do so after transaction execution, but may also do so after, for instance, finding that it does not share any instrument classes in common with a vendor. A peer application has the option of closing the session (but may just return an error if desired) if the application makes an invalid call, such as calling a method with an instrument class that is undefined or with an instrument class that it did not return as the result of `get-Remote-Available-Instruments`.

The application should "lock" the appropriate instrument instances upon closing a session. The results of all instrument negotiation, protocol negotiation, and protocol selection are forgotten upon close of the session, although the wallet architecture may be extended to include a feature to support caching of such information in the future.

After closing all sessions, the user may decide to shut down the wallet, at which point all wallet components save any unsaved information to persistent storage.

## 6.0 Related Work

*Java Wallet / Java Electronic Commerce Framework.* [5] The Java Wallet is not purely client-driven. In Sun's Java Electronic Commerce Framework, electronic commerce operations are initiated when a merchant server sends a Java Commerce Message (JCM) to a client's web browser. A JCM has a MIME-type of `application/x-java-commerce`, and the client's web browser will invoke the Java Wallet once the JCM message is received [9]. By convention, a vendor should not send a JCM to a client unless the client clicks a "PAY" button on a form on the vendor's web site. However, there is nothing preventing a vendor from sending a JCM to the client and invoking their wallet in response to the client simply visiting a page on the vendor's web site. These JCMs may be generated by CGI (Common Gateway Interface) scripts or servlets on the server-side, and sent to the client to invoke the Java Wallet. Alternatively, a JCM may also be generated by an applet that is downloaded to the client browser. In this case, the applet can make a method call on the JECF installed on the client to invoke the Java Wallet. Another way in which a vendor can invoke a user's Java Wallet in an unsolicited fashion is by sending them HTML email with such an applet embedded within it. Users that run HTML email readers that are integrated with their web browser, such as Netscape Navigator or Microsoft Internet Explorer and also have the JECF installed can have their wallet automatically invoked upon reading unsolicited email received from vendors. In order to render the HTML email message, the HTML will be parsed and the Java Virtual Machine will be started to render the applet the vendor embedded in the email. In this scenario, the applet on the page will start executing, generate a JCM, and make a `JECF.startOperation` method call passing the generated JCM as a parameter to invoke the user's Java Wallet.

Vendors can use these mechanism to urge customers to make impulse purchases simply by invoking an end-user's wallet when the end-user visits their web site or receives email from them. Customers may resent this feature since it gives vendors the ability to "take the customer's wallet out of his pocket." In our architecture, a user must *explicitly* launch the wallet to make a payment; this allows the user to make the payment when he or she pleases, and not when the vendor decides it is the appropriate time to pop-up the user's wallet.

Besides being client-driven, our digital wallet architecture supports the notion of a session while the JECF does not. In our model, once a client decides to initiate a session with a vendor, state information may be accumulated throughout the session, and multiple operations may take place during a single session. After initial instrument and protocol negotiation, the negotiated selections are retained as part of the state information in the session, and making additional payments thereafter does not require re-negotiation for each payment operation between the wallet and vendor. This mechanism allows us to implement lightweight instruments and protocols to execute micro-payments. However, in the JECF model, a separate JCM from the vendor is required to execute each commerce transaction, and all of the arbitration, negotiation, and selection may need to be done for each JCM. This can make the execution of micro-payments more costly in the JECF model.

*Microsoft Wallet. [6]* The Microsoft Wallet is composed of two Active/X controls: an Address Selector control, and a Payment Selector control. This model is also not purely client-driven, since vendors embed these controls on web pages on their web site to prompt the user to make a payment. Furthermore, the selection of the protocol is not client-driven. Within an HTML tag passed as a parameter to the Payment Selector control is an "accepted types" string which contains a list of (instrument class,protocol) pairs that are acceptable to the vendor. Upon choosing an instrument class, the accepted types are scanned from left to right searching for the first occurrence of the selected instrument class, and the corresponding protocol is chosen. Since the vendor orders the accepted types string, the vendor has the ability to choose a protocol that is advantageous to itself and disadvantageous to the client. For example, the vendor may choose a protocol that may lower its transaction cost, but that may take a longer time to execute over the network, costing the client more in network access charges. In our architecture, the Protocol Negotiation Protocol objects running on both the wallet and the vendor negotiate on the protocol to be used for a selected instrument class, and the client is not locked into using the fixed algorithm hard-coded within the Payment Selector control.

*IBM Generic Payment Service. [15]* IBM Zurich Research proposes a Generic Payment Service as part of the SEMPER (Secure Electronic Marketplace for Europe) project. The service gives the user the ability to have multiple "purses," each representing a different payment system. The concept of a "purse" in the contexts of the Generic Payment Service roughly corresponds to a combination of an instrument instance and an associated protocol.

*Shopping Models. [10]* The Shopping Models Architecture formalizes many different customer, merchant, and payment service interactions in terms of order, payment, and delivery event handlers. Our wallet architecture addresses payment in the context of Shopping Models. The wallet and vendor controllers, for example, expand on the interfaces that the `CustomerPayment` and `MerchantPayment` event handlers present in the context of that work.

*U-PAI (Universal Payment Application Interface). [11]* U-PAI proposes a standard interface to multiple payment mechanisms. A U-PAI `AccountHandle` fits into our architecture as a combination of an `Instrument` and a `Protocol` object since the `AccountHandle`'s interface provides methods to access instrument data, such as an account balance, as well as methods to execute a payment such as `startTransfer`.

*JEPI (Joint Electronic Payments Initiative). [12]* JEPI was a joint initiative between the W3 Consortium and CommerceNet whose goal was to develop a payment selection protocol as an extension to HTTP. JEPI's UPP [13] protocol is an example of a Protocol Negotiation Protocol in our architecture. Assuming a Communication Manager that is capable of sending HTTP messages (with the appropriate PEP extensions), UPP may be implemented within our architecture as a Protocol Negotiation Protocol.

*NetBill. [14]* In the NetBill protocol, payment is guaranteed to happen atomically with the delivery of goods by involving a trusted third party. To implement the NetBill payment mechanism in our architecture, the end-user wallet, vendor, and trusted third party applications would execute a NetBill payment protocol that would interact with NetBill money instrument objects.

*SET. [1]* SET is a secure electronic transaction protocol developed jointly by Visa and Mastercard. As mentioned previously in examples throughout the paper, SET is a protocol object within our architecture that can be used to make payments, as well as execute other operations defined in the SET protocol, with Mastercard and VISA credit card instrument classes.

*GEPS (Generic Electronic Payment Services). [7]* In the context of GEPS, the Instrument Manager takes advantage of Capabilities Management (CM) and Method Negotiation (MN) services. The Protocol Manager takes advantage of Capabilities Management (CM), Method Negotiation (MN), and Transaction Management (TM) services. The User Profile Manager takes advantage of Preferences Management (PM) services.

## 7.0 Conclusion

We propose a new generalized digital wallet architecture that is extensible, symmetric, non-web-centric, and client-driven. This architecture not only is extensible enough to inter-operate with multiple instruments and protocols as some existing wallet architectures are, but also incorporates other desirable features of a digital wallet architecture in a comprehensive way. In particular, the SWAPEROO wallet architecture also

- Factors out symmetric infrastructure and interfaces that may be common among end-user, vendor, and bank wallets,
- Generalizes to operating environments beyond the WorldWide Web, such that digital wallets can be developed for "alternative" devices such as PDAs and smart cards without re-inventing a new design for the wallet, and
- Ensures that the client is the proactive party in the payment phase of the shopping interaction.

Our proposed generalized digital wallet architecture has been implemented in Java and C++. Using our implementation, we built a digital wallet application for the PalmPilot personal digital assistant, and a vendor application that interfaces with a vending machine; the details of that particular implementation using the SWAPEROO wallet architecture is described in [17]. Complete Java and C++ APIs for the architecture are available at `http://www-db.stanford.edu/~daswani/wallets/`.

## Acknowledgements

We would like to thank Craig Mudge for participating in several discussions with us which helped contribute to ideas in this paper. We also wish to thank the anonymous referees for their helpful comments which aided in various revisions of this paper.

## References

[1] SET Secure Electronic Transaction (TM) LLC. SETCo website: http://www.setco.org/.

[2] CyberCash. CyberCash Home Page. CyberCash website: http://www.cybercash.com/.

[3] DigiCash. DigiCash: Solutions for Security and Privacy. DigiCash website: http://www.digicash.com/.

[4] Digital Equipment Corporation. MilliCent. MilliCent website: http://www.millicent.digital.com/.

[5] Sun Microsystems. Java Commerce Home Page. JavaSoft website: http://java.sun.com/commerce/.

[6] Microsoft Corporation. Microsoft Wallet. Microsoft wallet website: http://www.microsoft.com/wallet/.

[7] Alireza Bahreman. Generic Electronic Payment Services. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, 1996.

[8] Alireza Bahreman and Rajkumar Narayanaswamy. Payment Method Negotiation Service. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, 1996.

[9] Java Commerce Messages White Paper. Sun Microsystems website: http://java.sun.com/products/commerce/docs/whitepapers/jcm_whitepaper/jcm_whitepaper.html.

[10] Steven P. Ketchpel, Hector Garcia-Molina, and Andreas Paepcke. Shopping Models: A Flexible Architecture for Information Commerce. In *Proceedings of the Fourth Annual Conference on the Theory and Practice of Digital Libraries*, 1997. At http://www-diglib.stanford.edu/cgi-bin/WP/get/SIDL-WP-1996-0052.

[11] Steven Ketchpel, Hector Garcia-Molina, Andreas Paepcke, Scott Hassan, and Steve Cousins. UPAI: A Universal Payment Application Interface. In *USENIX 2nd Electronic Commerce workshop*, 1996.

[12] W3C Joint Electronic Payments Initiative (JEPI). W3C website: http://www.w3.org/ECommerce/Overview-JEPI.html.

[13] D. Eastlake. Universal Payment Preamble Specification. W3C website: http://www.w3.org/ECommerce/specs/upp.txt.

[14] B. Cox, D. Tygar, and M. Sirbu. NetBill Security and Transaction Protocol. In *First USENIX Workshop of Electronic Commerce Proceedings*, 1995.

[15] J.L. Abad-Peiro, N. Asokan, M. Steiner, M. Waidner. Designing a Generic Payment Service. *IBM Systems Journal Vol. 37 No. 1*, 1998.

[16] T. Goldstein. The Gateway Security Model in the Java Electronic Commerce Framework. In *Proceedings of the Financial Cryptography First International Conference, FC '97, 1997.*

[17] N. Daswani, D. Boneh. Experimenting with Electronic Commerce on the PalmPilot. [*preprint*]

# The Eternal Resource Locator: An Alternative Means of Establishing Trust on the World Wide Web

Ross J. Anderson, Václav Matyáš Jr., Fabien A.P. Petitcolas
*University of Cambridge Computer Laboratory*
*Pembroke Street, Cambridge CB2 3QG, UK*
{rja14, vm206, fapp2}@cl.cam.ac.uk; http://www.cl.cam.ac.uk/~{rja14, vm206, fapp2}

## Abstract

Much research on Internet security has concentrated on generic mechanisms such as firewalls, IP authentication and protocols for large scale key distribution. However, once we start to look at specific applications, some quite different requirements appear. We set out to build an infrastructure that would support the reliable electronic distribution of books on which doctors depend when making diagnostic and treatment decisions, such as care protocols, drug formularies and government notices. The integrity, authenticity and timeliness of this information is important for both safety and medico-legal purposes. We initially tried to implement a signature hierarchy based on X.509 but found that this had a number of shortcomings.

We therefore developed an alternative means of managing trust in electronic publishing. This has a number of advantages which may commend it in other applications. It does not use export-controlled cryptography; it uses much less computation than digital signature mechanisms; and it provides a number of features that may be useful in environments where we are worried about liability. We also present our intermediate solution – the first ever large scale deployment of one-time signature systems. The move to one-time signatures enabled considerable simplification, cost reduction and performance improvement. We believe that similar mechanisms may be appropriate for protecting other information that changes slowly and remains available over long time periods. Book and journal publishing in general appear to be strong candidates.

## 1 Introduction

Medicine is one field in which serious attempts are underway in a number of countries to build large-scale decentralised trusted systems over the Internet to support a number of aspects of patient care, administration and research. Medical informatics has made unique contributions to the general pool of security know-how, and as medical practice is highly decentralised, many of these lessons may be applicable to Internet applications in general. Examples include the following.

- Conventional security policy models like Bell-LaPadula and Clark-Wilson do not work well in medical telematics, as they assume that security administration is centralised. Medicine is a business in which the field operatives are the most highly trained, the most trusted by the clientèle, and in most countries are also legally burdened with the duty to make access control decisions [Hor97]. This has led to the development of security policy models that locate the control in the leaves rather than in the root of the access control structure [And96]
- An attempt by the UK government's Department of Health to introduce a conventional X.509 certificate structure [Zer96] was opposed by the medical profession. There was not just an ethical objection to the escrow features of the proposed key management protocol [JMW96]; it had also become clear that a central trusted third party could not cope with a health service whose approximately one million employees are

spread over 12,000 separate organisations.

- In addition to the engineering costs of centralisation, there was also the principle that electronic trust structures should mirror those in existing professional practice. This principle was first enunciated by Alexander Rossnagel, a German lawyer, who feared that certification authorities would deprive notaries of their income [Ros95]; it has since been adopted by the UK government and the medical profession as one of the principles by which disputes over electronic privacy issues are to be resolved [NHSE96].
- Governments attempted to solve security problems in Germany and Austria by introducing smartcards as access tokens for both patients and healthcare professionals. Despite careful government studies of the likely social consequences [BSI95], it has turned out that these cards have had a centralising effect that many professionals find intolerable [HW97].

The previous work that directly concerns us is Wax [Wax97a]. This is a proprietary hypertext system used for medical publishing; its goal is the secure and timely electronic distribution of information used to support clinical practice, such as treatment protocols and drug formularies. It will also be used for government circulars ranging from purely administrative information such as advice on coping with the Y2K bug to notices of newly discovered adverse drug reactions; and for local information such as hospital waiting lists.

Wax is already used in several health trusts in the UK for providing a mixture of trust-specific and general information. It is also used in the US for delivery of medical knowledge relating to HIV and AIDS by Intelligent Medical Objects, Inc. (Northbrook, Illinois, USA). There are clear safety and medico-legal reasons why the authenticity, integrity and timeliness of the information it distributes should be protected, and a project was undertaken during 1996-8 to design and implement this. That project is described in [Wax97a], but we will describe it here briefly so this article is self-contained.

## 2 Wax

The information in Wax is structured hierarchically with the levels being a shelf (owned by a publisher) containing books (each owned by an editor) consisting of chapters (each owned by an author). Thus if a primary care physician wants the latest advice on the conditions under which a patient with gout should be referred to hospital, he will draw from the Wax shelf the book on rheumatology and consult the chapter on gout. This chapter will have been written by a leading specialist and will be updated as necessary (typically every few years); the editor's job is quality control, principally choosing the experts and ensuring proper peer review.

As a solution was sought rapidly, an initial attempt at protection using digital signatures was undertaken using materials ready to hand – SHA, RSA, and X.509 [X509]. This decision was influenced by the fact that RSA with exponent 3 has just been accepted as the European standard for healthcare signatures. The X.509 hierarchy was founded on a Wax-root key, whose public component is embedded in the Wax browser software; Wax-root signatures certify keys of medical publishers (the Wax-centre for treatment protocols, the British Medical Journal, the Department of Health, individual hospital trusts, etc.) and the publishers in turn certify the keys of editors and authors.

As we did not know the optimum granularity of the signed objects, and had an operational requirement to open already cached books quickly, we also implemented a secondary protection mechanism whereby the book index contained (invisibly to the human reader) the SHA hashes of each chapter, and each shelf catalogue similarly contained hashes of book indexes. Thus a given book can be verified by means of its editor's signature, and also by reference to the publisher's catalogue. There is also considerable machinery to deal with trusted distribution of the

Wax software, trusted updating of local catalogues, and trusted collection of public keys from authors, none of which concern us here.

## 3 The first lessons

The principal lesson that we learned from this exercise was that the X.509 mechanisms are not really suitable for publishing. This realisation started to dawn when we had to decide on the longevity of publisher's keys. Assuming that users' keys would last three years, why not make the publishers' last five? But what would happen once a publisher's key was more than two years old, and thus unable to issue a certificate of three years' duration for an author? Would he have to refresh it, or acquire another one?

Many further complexities arose. For example, what does revocation mean in the context of book publishing? If an author fails to pay his annual fee to the local CA, will all his books magically vanish from all library shelves? What if a lawsuit is then brought in which a party relies on one of them? And what if revocation mechanisms were used maliciously as an instrument of censorship?

'Planned obsolescence' may make sense in software publishing, and in the banking world it is quite natural to use X.509 certificates in SET where both public and private keys have a lifetime of two years, as this simply replicates the existing trust structure of magstripe credit cards. However this approach is not appropriate in publishing, where objects are long-lived. Book copyright in the EU countries is now for a period of 70 years after the author's death.

The conclusion to which we were unexpectedly driven by the Wax project was that our secondary trust mechanism – namely, a tree of hashes in which chapters are hashed into a book and books into a catalogue – should in fact be the primary mechanism, while the X.509 signature mechanisms, which we had anticipated would provide the primary protection, are relegated to a number of secondary and specialist roles. The basic functionality can be seen in the following diagram (Fig. 1).

The question that we were naturally led to ask was whether catalogue-based trust had other natural applications than medical publishing, and what extensions of it might be appropriate. Our conclusion is that it gives a much better solution to some problems currently tackled using public-key cryptography, ranging from assuring the authorship of applets through enabling web authors to protect themselves from libel actions; in general, we can adopt the Wax mechanisms to provide a simple and robust set of mechanisms to authenticate the content of world wide web and other hypermedia systems, which fits well with the actual trust model that people have for published content.

We will now describe a set of proposed extensions to HTML that explain what we have in mind and make clear what can be achieved with it.

## 4 The Eternal Resource Locator

Trust, in the electronic world, is based on binding real-world assurances and/or relations to their electronic representation. This is expensive, and so in order for the trust transfer mechanisms (such as electronic signatures) to give maximum value, one should perform such bindings infrequently (but well). This is true for establishing a root of trust (e.g. top level Certification Authorities) and also for lower level entities. For example [CM97], it is a bad idea to bind keys and access rights to principals like this:

$$\text{key} \rightarrow \text{principal} \leftarrow \text{capability}$$

as this involves two bindings between the real world and cyberspace. We should rather build systems like this:

$$\text{principal} \leftarrow \text{key} \leftarrow \text{capability}$$

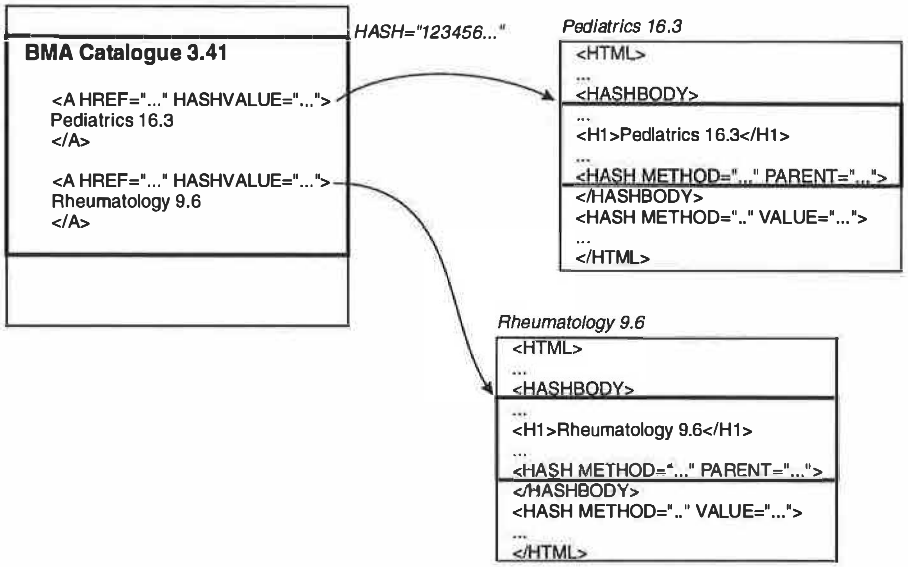


Figure 1: Publishing medical information. The publisher issues a catalogue every few months which lists all titles published, together with their hash values. The hash of the catalogue has to be distributed a trusted way, by being published in a paper journal, and signed using a long-term key.

Thus, when designing trust structures in general, we will try to have a small number of root keys or other authenticator values that can be made well known by out-of-band mechanisms, and derive the rest of the structure directly from these. The cleaner the structure, the better for a number of reasons, including both cost and robustness.

With trust based on hash trees, the root is quite simply the root of the tree; in the case of Wax, the hash of the current Wax catalogue. This can be assured by a variety of means (currently signature with the Wax-centre key and publication in the British Medical Journal). However, once we have gone to the trouble of certifying this root, we want all the pages in the publishing hierarchy to be checkable from it. We will now describe how this can be done using a simple extension that does not upset existing browsers, yet can be implemented either as a suitable applet or as part of a proxy service such as a hospital firewall. (The former is preferable as it is easier to implement 'untrusted' highlights, e.g. when a Wax chapter is retrieved as the result of a keyword search.)

## 4.1 Basic (static) mechanism

It would be clumsy to insist on the signature of whole web pages, so instead we propose to use HTML elements [HTML] to define the borders of the hashed section of the document as well as other features of the hashing mechanism:

- The HASHBODY element denotes the hashed section of an HTML document. All the text and HTML code in this section will be hashed with various hashing algorithm specified in the HASH element.
- The HASH element is an extensible container for use in identifying hash document information. It has three main functions: define the hashing algorithm used, store the corresponding hash value and link the element to a parent. The HASH element should be used both inside the HASHBODY section and outside; the purpose is to bind the protected section to its hash and its parent.

There can be as many **HASH** elements as hashing algorithms. Attributes of the **HASH** element:

  - **METHOD** specifies the hashing algorithm. A number of algorithms may be used in parallel in order to give reassurance against cryptanalytic progress;
  - **VALUE** specifies the value of the hash;
  - **PARENT** provides a pointer to another HTML document, called parent and specified by its URL. This enables a browser that wishes to check the page's integrity to follow the hash chain to a suitable root. The name of the root may be given for performance reasons. If there is no parent (i.e. the document is a root) the attribute should not be omitted but instead should be set to NO.
  - The **URL** attribute optionally specifies where the page normally lives, and can provide basic protection against attacks involving the copying of pages to false hosts. Care is needed not to get entangled with HTTP server's interpretation of URLs like `http://www.foo.com/` and `http://www.foo.com/index.html`.

- We also propose to add new attributes to the existing **A** element. This will allow to define the hash of the resource specified by the **HREF** attribute. It is assumed that the linked resource is different from the current resource as the hashed body cannot include the hash value itself. Obviously, these can be two different fragments of the same web page.

  **HASHMETHOD**, **HASHVALUE** and **HASHPARENT** have the same meaning as the **METHOD**, **VALUE** and **PARENT** attributes of the **HASH** element.

Simplified, the way to protect part of the web page will look like:

```
<HTML>
...
<HASHBODY>
...
The examination results for the second MB
degree examination are as follows:
...
...
<HASH
URL="http://www.med.abc.ac.uk/examresults"
METHOD="SHA-1"
PARENT="http://www.cert.bma.org.uk">
<HASH
URL="http://www.med.abc.ac.uk/examresults"
METHOD="TIGER"
PARENT="http://www.cert.med.ac.uk">

</HASHBODY>

<HASH METHOD="SHA-1" VALUE="12345678..."
PARENT="http://www.cert.bma.org.uk">
<HASH METHOD="TIGER" VALUE="987654321..."
PARENT="http://www.cert.med.ac.uk">
...
</HTML>
```

One of the URLs that refer to this page might look something like:

```
...see <A
HREF="http://www.med.abc.ac.uk/examresults"
HASHMETHOD="TIGER" HASHVALUE="987654321..."
HASHPARENT="http://www.cert.med.ac.uk">here
</A>for the list of candidates who have
satisfied the requirements for the degrees
of MB and BS.
```

Checking a hash involves computing the hash value on all the bytes of an HTML document between the hash-input border tags and comparing the HTML document's URL against the value specified within the hash-input. This value is then verified against the value held in the reference in the parent document.

We call this URL-with-hash combination an ERL or 'eternal resource locator' as it makes static objects unique for ever. Dynamic objects are slightly more complex.

### 4.2 Dynamic pages

If we used hash functions alone, then this would limit us to material that was available and known when the last issue of the catalogue was published. Almost all published medical information is of this nature; it changes relatively slowly owing to safety protocols that insist on thorough peer review and validation. However, there is a demand for a small number of dynamic pages in the system holding 'hot' news such as recently advised drug side effects and other safety notices, or operational data such as test results. What we do not want to do is say something like 'for recent notices on test results $X$, look at URL $Y$ for a message signed by a key certified to belong to $Z$' as this would suddenly involve reliance on a second root.

The effect would be that the referenced information was no longer part of the same trust structure, introducing complexity and making liability potentially uncertain. We therefore want to say something like 'for recent notices on test results $X$, look at URL $Y$ for a message signed by a key whose hash is $Z$'. We will now describe the details.

The owner of the dynamic page creates a signature keypair and embeds the public key as follows:

```
...see <A
HREF="http://www.med.abc.ac.uk/bloodtest"
HASHMETHOD="TIGER" HASHVALUE="987654321..."
HASHPARENT="http://www.cert.med.ac.uk">here
</A> for today's blood test results for the
Fisher medical practice...
```

which refers to this:

```
...
<HASHBODY>
...
The blood test results for the Fisher practice
on 22/8/98 are as follows:
...
<HASH
URL="http://www.med.abc.ac.uk/bloodtest"
METHOD="SHA-1"
PUBLICKEYVALUE="ABCDEF01234.......89ABC"
ALGORITHM="RSAE3"
PARENT="http://www.cert.bma.org.uk">
<HASH
URL="http://www.med.abc.ac.uk/bloodtest"
METHOD="TIGER"
PUBLICKEYVALUE="ABCDEF01234.......89ABC"
ALGORITHM="RSAE3"
PARENT="http://www.cert.med.ac.uk">
</HASHBODY>
<HASH METHOD="SHA-1" VALUE="12345678..."
PARENT="http://www.cert.bma.org.uk"
SIGNATURE-VALUE="FEDCBA987..........76543"
ALGORITHM="RSAE3">
...
```

The reason that the public key's presence is not made clear in the parent page is to preserve bandwidth (keys are relatively large) and because we could find no reason why someone when clicking on a link should know in advance whether it is statically or dynamically protected. It also makes the implementation simpler.

Note that although we are using public key cryptography, we have no need of an X.509 certification mechanism. All the trust links created by the public keys are local and transient. So there is no need for long-term secrets; everything gets suddenly much simpler, more manageable and more exportable.

## 5 Other applications

Ignoring dynamic links for the moment, the trust structure naturally supported by ERLs has an interesting and, from any publisher's point of view, highly desirable property: that you cleanly distinguish material of which you approve and in which you expect your readers to place some reliance. This may seem trite but is a growing concern, as in the laws of many countries a defamation suit may be brought against anyone involved in the distribution of a contested statement and not merely the author.

In the UK it is normal for libel litigants to sue and attempt to enjoin the distributors of newspapers and magazines with which they have taken issue; recently the Nottinghamshire County Council issued lawyers' letters and injunctions against a number of people who had links on their home pages to leaked copies of a report on satanic child abuse that the council considered to be its copyright [Notts].

So putting a link on one's home page can be dangerous; the controller of the referenced page might introduce controversial material and one could be sued. The implications in medicine include, for example, that a hospital which carelessly referenced a drug company's information page could find its standing in a negligence case substantially altered; equally serious consequences could follow elsewhere.

So the general use of ERLs rather than URLs would often be prudent practice, as the failure of a followed link to authenticate will indicate that it has been changed since the author of the link last consulted it, and he can thus in no way be held liable for its contents.

Other applications will typically arise where a publisher owes some particular duty of care, and we suggest some examples below.

## 5.1 Public keys with multiple accreditors

It is quite common to assign a person a role whose performance depends on using a role key. However, we can have multiple parties having to approve assignment of such role. This is common in banking, where transactions over a certain amount typically have to be approved by more than one officer, but may be delegated on a limited basis. At present, special key management standards are being developed for banking by ANSI, as multiple signatures are not supported by X.509.

A similar problem arises in medicine, where the signing key used by (say) a doctor on a six month assignment in a hospital would not wish to use his long term personal signing key (hospital systems are often mutually incompatible and quite insecure) but would instead use a key that was signed both by his own personal long term key (in an off-line operation) and the hospital. This dual signature signifies that both the doctor and the hospital accept their joint liability for malpractice suits; it should also be possible for either of them to revoke the key when the relationship is terminated. This multiple revocation requirement is quite a complicated problem if one tries to implement everything as extensions of X.509.

Neither are medicine or banking the only applications in which dual control is required. Almost every substantial organisation has its own rules and procedures for managing dual control. Sometimes these rules may be hidden, as with military intelligence organisations who do not wish to reveal which officers have actual power; at other times, concealment is forced, as with the system of EU grants under which each grant receiving organisation (such as a university) must designate one 'official signatory' and the European Commission refuses to take any interest in the procedures that may lie behind this person's use of his official signature. How can we then support realistic trust models using either catalogue based or public key based trust mechanisms?

In the old days of paper-based banking systems, the custom was for each bank to print several hundred copies of a 'signature book', which contained the specimen signatures of its managers together with a set of rules defining, for example, which combinations of signatures were required on a letter of credit over $10m. These books were then posted to its correspondent banks.

Our catalogue based trust mechanisms can provide an electronic implementation. In its simplest form, the company authenticates at regular intervals a set of public keys suitable for appropriate purposes and makes them available via web pages bound to the relevant trust trees.

This can even be done if need be in real time; we are experimenting with a mechanism whereby flexible links can be created on request to authenticate a key for a particular purpose. The example in the following diagram (Fig. 2) is where a company lawyer wishes to create a one-time key to conclude a property transaction that involves both internal certification (from his superior officer and the company's CEO) and an external land and property agency.

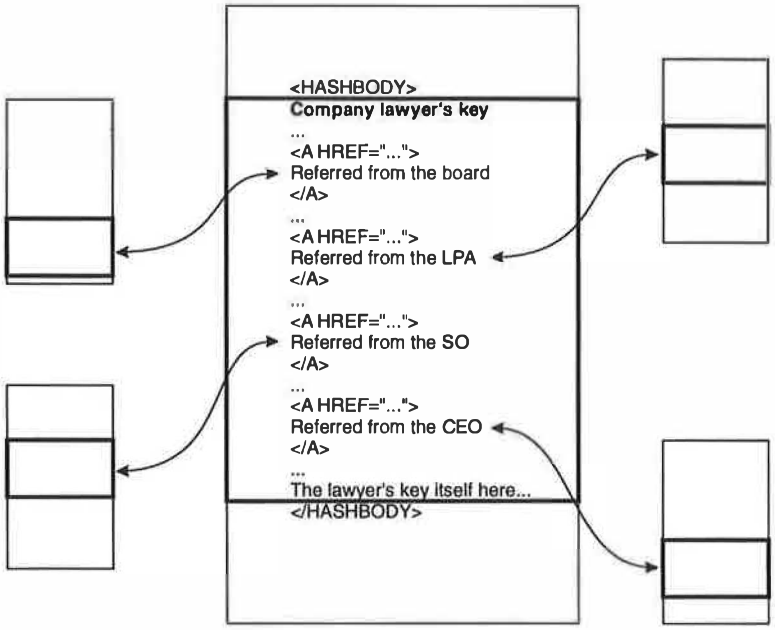


Figure 2: Publishing key information. Multiple accreditors are referring to the key that is valid if and only if all self-contained required links exist and the key-page hash value has not changed.

The effect of this mechanism is that CA functions can be performed on a one-off basis by various people and organisations as they are needed – a flexibility that is still critically lacking in X.509.

## 5.2 Timestamps

Time stamping services such as Surety's [HS91] are another example of a hash tree. In this case there is a mechanism for recomputing the tree and reliably publishing the hash every second, thus allowing rapid generation of an existence proof for a document.

We hope that formats for the inclusion of timestamps and other such evidence within the ERL structure and within HTML generally can be developed that is acceptable and useful to all parties.

## 5.3 Other considerations

The ERL idea also brings back the 'natural' concept of trust in a broader sense — it highlights the point of who trusts the certified entity, not the current notion of whom the certified entity trusts (or is forced to 'trust').

The ERL system requires reliable support of computing and checking hash values, in some cases supported by time-stamping services. It is also desirable to have the supportive mechanisms embedded within the existing computer environment (hardware, OS, Web browser or so).

We believe that our system is easily applicable on current Web platforms. The Web browser or OS should include features to compute the hash value of an arbitrary input (software distribution file, etc.) and perform a check on the hash of a loaded HTML document. It would be advisable to support several different hashing algorithms to avoid any future problem or failure. Also, hashes should be stored within the bookmark file of the browser, where they could provide some level of document change control exploitable for cache management, triggering warning mechanisms where relevant, etc.

# 6 The current version of Wax

The success of Wax led to interest from other countries. A US software company, Intelligent Medical Objects, Inc. (Northbrook, Illinois) decided to adopt the system as its preferred method for delivery of medical knowledge relating to HIV and AIDS. This would involve distribution of the browser software to over 300,000 physicians and other carers in the USA, and led immediately to a dilemma.

The owner of the RSA patent, RSA Data Security Inc., insists on a royalty that is a function of the sale price of software incorporating its technology, and which in the case of software distributed for free has a minimum value of $5.00. The Wax project having been funded by charitable money, research grants and volunteer labour, was not in a position to pay $1.5m as the price of entry to the USA.

This compelled the Wax project to revisit the cryptography issue. Another team was put together and we took a look at the design and trust issues. We found that we could achieve the same goals as before, and even more simply, by using one-time signatures instead of RSA. Necessity had truly become the mother of invention.

## 6.1 Application of one-time Signatures

In this subsection we present our solution based on one-time signatures. First we will recall briefly the ideas behind one-time signatures and then detail our implementation.

### 6.1.1 One-time signature basics

One-time signature scheme was first introduced by Lamport [DH76, Lam79] and more efficient schemes have been proposed since then [Mer87, Mer89].

We will assume here that the hash function $h$ produces $l$ bits and the message digests to be signed are $n$-bit long. The first step involved is the creation of a key-pair which will be used to sign a file only once; for this purpose, two arrays $\mathbf{X}$ and $\mathbf{Y}$ are generated. The first one contains $N = n + \lceil \log_2 n \rceil$ truly random $l$-bit-numbers $x_1 \cdots x_N$, and the second contains the hash values of these numbers, that is, $y_i = h(x_i)$. By definition the public key is: $K = h(\mathbf{Y})$.

The second step is to compute the signature of a file $f$ whose hash is noted $H_f$. The signature of $f$ is simply an array $\mathbf{S}$ whose $N$ components are:

$$\begin{cases} s_i = (1 - h_i)x_i + h_i y_i & \text{if } 1 \leq i \leq n, \\ s_i = (1 - c_i)x_i + c_i y_i & \text{if } n < i \leq N, \end{cases}$$

where the $h_i$'s are the binary digits of $H_f$ and the $c_i$'s the binary digits of a checksum $c = \sum_{i=1}^{l} h_i$. This checksum prevents attacks in which an opponent could produce a file $f'$ such that all the '1' in $H'_f$ are also in $H_f$ but some '0' in $H_f$ have been replaced by '1'. Once the signature is generated, the private key $\mathbf{X}$ should be destroyed.

Given $f$, $\mathbf{S}$ and $K$, verifying the signature implies: compute $H_f$, $c$, construct an array $\mathbf{Y}$ such that:

$$\begin{cases} y_i = (1 - h_i)h(s_i) + h_i s_i & \text{if } 1 \leq i \leq n, \\ y_i = (1 - c_i)h(s_i) + c_i s_i & \text{if } n < i \leq N, \end{cases}$$

and check that $h(\mathbf{Y}) = K$.

## 6.2 The current implementation

In the current version of Wax we apply the above one-time signature at the catalogue level. Each catalogue contains the hashes of all relevant books which can then be simply authenticated via the catalogue. We also link our catalogues together. We include in each signed catalogue seven public keys with which further editions of the catalogue, and other material from the same publisher, can be authenticated. This means that the chain of trust is broken if a user skips too many updates; in that case, he has to verify the public key of the new update using the same out-of-band mechanisms employed when the system was initially loaded and which we describe below.

For the initialisation of the chain of trust the Wax-centre generates eight key pairs $(\mathbf{X}_1, K_1) \ldots (\mathbf{X}_8, K_8)$. The first catalogue $(C_1)$ contains the hashes of its associated files $f_{1,j}$ and the next public keys: $C_1 = \{h(f_{1,j})\}, K_2, \ldots K_8$. It is signed using the first private key: $\mathbf{S}_1 = s(C_1, \mathbf{X}_1)$. As mentioned above $\mathbf{X}_1$ is destroyed but the remaining $\mathbf{X}_i$'s are kept for the following distributions.

For the $i$th delivery a new key pair $(\mathbf{X}_{i+7}, K_{i+7})$ is generated and a new catalogue prepared:

$$\begin{cases} C_i &= h(C_{i-1}), \{h(f_{i,j})\}, K_{i+1}, \ldots, K_{i+7} \\ \mathbf{S}_i &= s(C_i, \mathbf{X}_i). \end{cases}$$

Including the hash of the previous catalogue(s) into the current one prevent from denial of content attacks. Again, $\mathbf{X}_i$, the private key used for signature, is destroyed.

In order to bootstrap the trust in the system, each user is required to verify the public key $K_1$ of this initial shipment. A number of channels are provided for this, which is tightly bound up with the problem of trusted distribution. Initial deployment is by means of a mass mailing of CDs (stuck to the cover of an appropriate medical journal); electronic distributions are also available with authentication provided by the available mechanisms (such as PGP signatures, published MD5 hashes in medical journals, and download using SSL from a 'secure' web site).

The version of Wax that used RSA and X.509 had some further mechanisms, that were involved with users registering public keys of their own to the system; the corresponding private keys were used to generate signatures on books generated locally (such as treatment protocols developed in an individual medical practice) and also to generate counter-signatures on catalogues which had been downloaded and verified (as an extra precaution against virus attacks and the like).

On reviewing this design we concluded that the local use of public key cryptography added little. A medical practice which is going to publish a locally developed treatment protocol will as a matter of basic safety submit it to a peer review process, and thus all publication either is intermediated or can easily be made so. As for virus attack, the use of local signatures really only adds a modest layer of 'security through obscurity' as a virus written after study of the Wax code could alter the local public key and, absent tamper resistant processors, there appears to be no way to stop this.

### 6.3 New books

In order to issue an update of one or more books, a publisher just has to create a new catalogue and include the hashes of the books in it, then generate a new keypair and include the public part. The catalogue is now signed and made available for download together with the new book.

A user can choose which books he wishes to download and update. He must first download a catalogue and verify its signature. Then he can start downloading all or some of the books in the catalogue. Once these books have been downloaded and their integrity checked, the local index of books is updated and a checksum retained locally using 3-DES encryption and a passphrase, signed using a new one-time key pair.

### 6.4 New publishers

New publishers can introduce themselves at any time by simply publishing a catalogue, making it available for download, and providing out-of-band mechanisms for verifying the initial public key. This, however, replicates the effort required for out-of-band verification.

A simpler solution, which we have implemented, is to designate a special publisher (known as 'Wax-Root') whose sole function is to introduce new publishers. Each Wax-Root catalogue assigns one (or exceptionally more than one) of the 7 public keys to the new publisher's first catalogue, and in this way we branch the authentication tree.

## 7 Conclusions

This work has provided a number of insights.

Firstly, the use of hash trees rather than certificate chains is appropriate for trust relationships that change slowly ($X$ is an employee of company $A$) or not at all (book $Y$ was published by company $B$). Public key certificates are less suited for such relationships, at least in their most common forms: the typical three year lifetime of a key in an X.509 certificate system is too short for such applications; while a private signing key with a 100-year lifetime would be hard to protect (indeed, even three years may be too long a period to protect a really valuable private key). Catalogue based trust is one way of escaping this difficult trade-off between the need for a long-lived public key and a short-lived private one.

Secondly, trust mechanisms built using hash trees are simple to implement, intuitive to use, cost little in performance terms, and need contain no export-controlled mechanisms such as asymmetric cryptography.

Thirdly, catalogue based trust has a very natural fit with the publishing industry's business model in a number of ways ranging from the need to be careful about libel to the fact that catalogues are used anyway. Publishing is no longer a matter of the manufacture and distribution of books and newspapers; much of electronic commerce is publishing in some sense or another, and even where the net is used to sell widgets its main function is the publication of price lists, product data, delivery schedules and other information that is most easily organised in the form of one or more catalogues.

Fourthly, catalogue based trust mechanisms can be used to compensate for the shortcomings of X.509-type systems, such as the failure to support multiple certification discussed above.

Fifthly, catalogue based trust is robust. Cross-links can be inserted easily, in that a given book might appear in its publisher's sales catalogue and also in its editor's CV. Thus the security failure of one or another of these documents will leave the reliability of the book in unchanged. Such resilience is harder to achieve using X.509.

These advantages have become apparent to us in the course of the Wax project. We have sketched how very simple extensions to HTML can make them available to the net generally. We invite the authors, owners and proponents of other protection mechanisms to come together and agree a syntax for dealing with such protection tags in a standard way. The goal is no less than 'trusted browsing' – and, as this work makes clear, the admirable work already done in this direction by protocols such as SSL and SET is only the first step. There are many more protection goals than the secure transfer of credit card numbers from customer to merchant, and we believe that ERLs will make a significant contribution.

We have also developed a mechanism based on one-time signatures to assure the authenticity and integrity of electronic books. Although our particular application was medical, and was driven initially by a requirement to avoid RSA Data Security's patent, many of the lessons learned are much more general. We believe that the mechanisms described here are suitable for any application in which we need to assure the authenticity of relatively stable digital objects over long time periods, such as cataloguing, notarisation and archiving; they are certainly much more suitable than current incarnations of X.509 with all their expiry date and other problems.

There were other, less tangible, benefits. Moving from an X.509 implementation to this one-time scheme was like a breath of fresh air. Almost all the complexity vanished – from ASN.1 and DER, through modular arithmetic, to all the tricks used to protect local signing keys from casual attack. It was found that signatures based on one-way functions could be explained simply to the medical personnel involved in the project, as well as to programmers with no background (or interest) in cryptography. This made progress several times faster than had been the case when the RSA version was implemented in late 1996. The consequences for user trust in the system should not be underestimated; neither should the reduced likelihood that a design or programming bug will be discovered and exploited in attacks.

There will be applications in which a mixture of the number-theoretic and hash-function-based approaches will remain attractive. A book on investment, for example, might have its trust based on the techniques described here, but contain embedded public keys based on number theory in order to authenticate online pages of current stock prices. The advantage of such a structure is that these public keys now become independent of X.509 or any other public certification hierarchy, which is highly desirable given the lack of robustness of such mechanisms and the political struggles to control them. Such flexible links from a catalogue-based trust structure to more volatile items could, we believe, accommodate most of the world of journal and magazine publishing within the overall structure described here.

One current thrust of our research lies in extending the Wax mechanisms from proprietary to open publishing systems; ERLs will lay the foundation for this. Other directions of research include the control of updates to cached documents; allowing a user to store the hash with bookmarks and to be informed of changes – either when subsequently loading the document or at update; and interactions with the considerable range of other protection primitives in the security literature (anonymous messaging, digital cash, micropayments, incremental integrity primitives, copyright marking mechanisms and so on).

## 8 Acknowledgements

We would like to thank Iain Buchan and Rudolf Hanka for presenting us with the original challenge of providing security features for the Wax system, and to Bruce Christianson and Bruno Crispo for discussions on trust bindings.

## 9 Availability

More information on Wax, as well as the Wax software itself, can be found at

`<http://www.medinfo.cam.ac.uk/wax/>`

## References

[And96] 'A Security Policy Model for Clinical Information Systems', RJ Anderson, in *Proceedings of the 1996 IEEE Symposium on Security and Privacy* pp 30–43

[BSI95] *'Chipkarten im Gesundheitswesen'*, Bundesamt für Sicherheit in der Informationstechnik, *Bundesanzeiger*, 4 May 1995

[CM97] 'Binding Bit Patterns to real World Entities', B Christianson, JA Malcolm, in *Security Protocols Workshop 97*, Springer LNCS v 1361 pp 105–113

[DH76] 'New Directions in Cryptography', W Diffie and M Hellman, *IEEE Trans. on Information Theory*, IT-22, November 1976, pp 644-654

[HS91] 'How to Time-Stamp a Digital Document', S Haber, WS Stornetta, in *Journal of Cryptology* v 3 no 2 (1991) pp 99–112

[HW97] 'Daten aus der Psychotherapie – auch bei uns bald eine Ware?' A von Heydwolff, T Wenzel, *Psychotherapie Forum* v 5 no 1 pp 17–25

[Hor97] 'Computers can be compatible with confidentiality' JS Horner, in *Journal of the Royal College of Physicians of London* v 31 no 3 (May/June 97) pp 310–312

[JMW96] 'A Proposed Architecture for Trusted Third Party Services', N Jefferies, C Mitchell, M Walker, in *Cryptography: Policy and Algorithms*, Springer LNCS v 1029 pp 98–104

[Lam79] 'Constructing digital signatures from one-way function', L Lamport, *Technical Report SRI-CSL-98*, SRI International, October 1979.

[Mer87] 'A Digital Signature Based on a Conventional Encryption Function', RC Merkle, *Proc. CRYPTO'87*, LNCS 293, Springer Verlag, 1987, pp 369-378

[Mer89] 'A Certified Digital Signature', RC Merkle, *Proc. CRYPTO'89*, LNCS 435, Springer Verlag, 1990, pp 218-238

[NHSE96] Press release, NHS Executive, 17th October 1996

[HTML] 'HTML 3.2 References Specification – W3C Recommendation', Dave Ragget, January, 1997

[Notts] *Standard citation suppressed for legal reasons*; it can be found easily as 'The JET Report' using Web search engines.

[RRBL96] 'SDSI – A Simple Distributed Security Infrastructure', RL Rivest, B Lampson, at `<http://theory.lcs.mit.edu/~cis/sdsi.html>`, v1.0 presented at USENIX 96 and CRYPTO 96, April 30, 1996

[Ros95] 'Institutionell-organisatorische Gestaltung informationstechnischer Sicherungsinfrostrukturen', A Roßnagel, in *Datenschutz und Datensicherheit* (5/95) pp 259–269

[SHA] 'Secure Hash Standard', National Institute of Standards and Technology, *NIST FIPS PUB 180*, U.S. Department of Commerce, May 1993

[Wax97a] 'Secure Books: Protecting the Distribution of Knowledge', R.J. Anderson, V. Matyáš Jr., F.A.P. Petitcolas, IE Buchan, R Hanka, in *Security Protocols Workshop 97*, Springer LNCS v 1361 pp 1–11

[Wax97b] 'Coherent Exchange of Healthcare Knowledge in Open Systems', IE Buchan, R Hanka, in *Studies in Health Technology and Informatics*, C.Pappas et al (Eds), Vol 43, Medical Informatics Europe 97, pp 821–824, IOS Press, 1997

[X509] 'Information technology – Open Systems Interconnection – The directory: Authentication framework', *ITU-T Recommendation X.509*, November 1993; (June 1997 E)

[Zer96] 'The use of encryption and related services with the NHSnet', Zergo Ltd., published as NHSE IMG document number E5254, April 1996

# Detecting Hit Shaving in Click-Through Payment Schemes

Michael K. Reiter
*AT&T Labs Research*
*Florham Park, NJ, USA*
reiter@research.att.com

Vinod Anupam    Alain Mayer
*Bell Labs, Lucent Technologies*
*Murray Hill, NJ, USA*
{anupam,alain}@research.bell-labs.com

## Abstract

A web user "clicks through" one web site, the referrer, to another web site, the target, if the user follows a hypertext link to the target's site contained in a web page served from the referrer's site. Numerous click-through payment programs have been established on the web, by which (the webmaster of) a target site pays a referrer site for each click through that referrer to the target. However, typically the referrer has no ability to verify that it is paid for every click-through to the target for which it is responsible. Thus, targets can undetectably omit to pay referrers for some number of click-throughs, a practice called *hit shaving*. In this paper, we explore simple and immediately useful approaches to enable referrers to monitor the number of click-throughs for which they should be paid.

## 1 Introduction

Though the emergence of full-scale electronic commerce on the World-Wide-Web is proceeding slowly, the web has been quickly and aggressively realized as an effective advertising medium. Indeed, advertising itself has become an important commodity on the web. The latest evidence of this fact is the growth of *click-through payments*, in which the webmaster of one web site $B$ pays the webmaster of another site $A$ for every referral that $B$'s pages receive from $A$'s. In other words, if a web user, when viewing one of $A$'s pages, clicks on a link in that page to one of $B$'s pages (in this sense, the user has "clicked through" $A$ to reach $B$), then $A$ is entitled to payment from $B$. $B$ runs a click-through payment program to motivate others to prominently display links to site $B$ and thus to increase the traffic that $B$ receives.

Due to the structure of the HTTP protocol, click-through payment schemes as implemented on the web today hold many opportunities for fraud. The referrer $A$ is given no way to verify that it is paid for every referral its pages give $B$. This allows $B$ to undetectably "forget" some referrals that it receives from $A$, a practice called *hit shaving*. Moreover, even if $A$ were able to detect that its referrals were shaved by $B$, it has no evidence to present to a third party to argue this fact. The attention paid to hit shaving in discussions of web advertising (e.g., [Kle98]), often in advertisements for the click-through programs themselves, suggests that hit shaving is a recognized and prominent problem in the click-through industry today. Moreover, the stakes suggest that fraud is likely: some click-through programs advertise surprisingly large payments (e.g., up to \$6 per click-through) and prizes based on click-throughs (e.g., one web site advertised a click-through contest in which first prize was a 1998 Corvette).

The purpose of this paper is to bring the problem of hit shaving to the attention of the technical community and to explore remedies to the problem. We first focus on solutions that can be immediately useful on the web today: we offer web constructions (i.e., ways to construct web pages and CGI scripts) that enable a webmaster to monitor how often users click through her pages to others, and to which pages they click. Moreover, these techniques require no cooperation or awareness by the sites to which the referrals are made, making them very effective. Though only heuristic in nature and not foolproof, these techniques can immediately offer webmasters greater ability to detect hit shaving by click-through payment programs. We then explore more ambitious approaches that, with the cooperation of click-through program providers, enables webmasters to monitor more precisely the number of click-throughs for which they should be paid, and to even obtain nonrepudiable evidence of these click-throughs from the target site. Even though this second set of approaches requires cooperation by the webmasters of the click-through payment programs, these webmasters need not be trusted, in the sense that their failure to cooperate is quickly detectable. All of the techniques that we propose are largely invisible to the web user, in that the user experiences nothing out of the ordinary as a consequence of these techniques. Moreover, these techniques work with off-the-shelf browsers today.

## 1.1 The problem

We begin with a brief overview of how click-through payment programs work today, focusing on their susceptibilities to fraud and the various concerns that shape our solutions. We draw this description from several prominent examples of click-through programs on the web. In the rest of this paper, we often speak of a web site and its webmaster (i.e., the person that controls the content it serves) synonymously.

A click-through agreement is set up when (the webmaster of) one site $A$, the *referrer*, applies for an account at the *target* site $B$ that is running a click-through payment program. This typically takes place by $A$ filling out a form for $B$ in which $A$ provides, for example, its address to which payment checks should be mailed. After establishing the account, $A$ is given advertising material in the form of Hypertext Markup Language (HTML) commands to include in its web pages, possibly along with accompanying images ("banners") to display on its web pages. These HTML commands include a hypertext link to the target site $B$; i.e., when a user views $A$'s page and clicks on this link, then the user's browser retrieves the referred-to page from $B$. In this sense, the user has "clicked through" $A$ to get to $B$. Typically $B$ maintains an account statement for $A$, so that $A$ can periodically visit site $B$ to see how many referrals $A$'s pages have made to $B$ (and thus the amount of money to which $A$ is entitled).

To understand the risks for fraud in this mechanism, we need to review the actual Hypertext Transfer Protocol (HTTP) messages exchanged during a click-through. This exchange is shown in Figure 1. The exchange begins when the user's browser retrieves a web page from site $A$, say **http://siteA.com/pageA.html**. This page contains a hypertext link to a page served by $B$, say **http://siteB.com/pageB.html**. $A$ included this link in **pageA.html** when it registered to participate in $B$'s click-through payment program. When the user clicks on that link, her browser retrieves **pageB.html** from $B$. $B$ can use a header of the HTTP protocol, the **Referrer** header, to determine the URL of the page that referred the user to site $B$. In this case, the **Referrer** field will indicate the URL of **pageA.html**, and so $A$ will be credited with the referral.

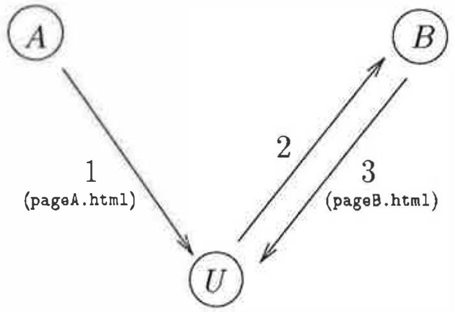


Figure 1: **A click-through**: User $U$ retrieves **pageA.html** from $A$ (message 1) and clicks on a link in it, causing **pageB.html** on site $B$ to be requested (message 2) and loaded (message 3).

The possibilities for abuse in this system should be evident. Since there is no communication to $A$ after the user clicks on the link to **pageB.html**, there is no way for $A$ to know how many referrals its pages give $B$. So, $B$ can just ignore the **Referrer** field of requests and thus fail to give proper credit to $A$; as described earlier, this is called *hit shaving* in the click-through payment industry. $B$ is also subject to abuses by $A$, e.g., if $A$ generates false requests to $B$ with **Referrer** fields naming **pageA.html**. In this way, $A$ can unfairly inflate the payment that it receives from $B$, and we will henceforth refer to such practices by $A$ as *hit inflation*. Like shaving, hit inflation is a recognized problem in the click-through payment industry. Many providers of click-through payment programs threaten cancellation of a referrer's account if hit inflation by an account holder is detected, but for obvious reasons providers do not typically reveal their methods for detecting hit inflation. It is likely that these methods are at least partially based on monitoring user IP addresses, and in particular detecting multiple requests from the same IP address or domain. Indeed, many click-through programs agree to pay only for "unique" referrals, i.e., referred requests from users at different addresses.

In this paper we treat only the problem of hit shaving. Nevertheless, the threat of hit inflation shapes the class of solutions that we are willing to consider. For example, one approach to detect hit shaving would be for $A$ to craft **pageA.html** so that its link purportedly to **pageB.html** is actually a link to a URL on site $A$; then, when the user clicks on that link, $A$ retrieves **pageB.html** from $B$ and serves it to the user. This enables $A$ to precisely know how many referrals it gives to $B$. However, this exacerbates the problem of detecting hit inflation, because it establishes a norm in which $A$ directly issues to $B$ all requests for which it should be credited. This hampers $B$'s ability to detect hit inflation based on user IP addresses. For this reason, we require that our solutions do not change the fact that the user's browser requests $B$'s pages directly from $B$.

### 1.2 Goals and assumptions

In the remainder of this paper, our goal is to enable site $A$ to monitor how many legitimate referrals it gives to site $B$, or more specifically, how many times $B$ receives an HTTP request from a user's browser for **pageB.html** with a **Referrer** header naming some URL on site $A$ (i.e., message 2 in Figure 1); we take this as the number of click-throughs for which $A$ should be paid. Note that this number includes any such request received by $B$, regardless of how $B$ responded to it (with **pageB.html** or with an error message), but it does not include requests that $B$ did not receive, e.g., because $B$ was down.[1] However, because $A$ cannot monitor exactly which messages $B$ receives, we must settle for solutions that enable $A$ to approximate this number. For reasons discussed in Section 1.1, it is difficult to monitor this number given the way that click-throughs presently work. Thus, our solutions modify how the click-through happens, but we allow them to do so only in a way as to impact traffic patterns, server load, and users' experiences as little as possible.

In forming our solutions, we assume that the user's browser is a frame-enabled and JavaScript-enabled off-the-shelf browser. We view the user's browser as a trusted-but-oblivious third party: we assume that it faithfully interprets the web pages (including JavaScript commands, when necessary) fed to it, but we do not assume that it has been modified in any way to support our approaches. In the JavaScript code segments included in this paper, we have allowed ourselves the full expressiveness of JavaScript 1.2, but versions for JavaScript 1.1, and in some cases JavaScript 1.0, can be formulated. The effectiveness of our JavaScript code segments has been verified using both Netscape Communicator 4.03 and Internet Explorer 4.0 over Windows 95 as the user's browser (subsequently abbreviated "NC4" and "IE4", respectively) and two Apache web servers on different hosts in different domains as sites $A$ and $B$.

## 2 Upper bounds on referrals

A first approach to detect hit shaving is to modify the click-through sequence to elicit a notification from the user's browser to the referring site $A$ when the user clicks the link to **pageB.html** in **pageA.html**. In this way, $A$ can monitor how many times users have clicked through **pageA.html** to **pageB.html** by monitoring how many such notifications it receives. In this section we show two approaches for achieving this, or more specifically for turning the exchange of Figure 1 into one that looks like Figure 2. As Figure 2 shows, once the user clicks on the link to **pageB.html**, a notification is sent back to $A$ (message 2) and then **pageB.html** is retrieved (messages 3,4). Neither of the methods we propose requires cooperation from site $B$, and both are invisible to the user.

Though effective, the methods of this section enable site $A$ only to record an *upper bound* on the number of referrals for which $A$ should be credited, not an exact count. The reason for this is that the notification sent to $A$ (message 2 in Figure 2) is an indication only that the user's browser will request **pageB.html**, not that $B$ has received this request. To see the importance of this distinction, the webmaster of site $B$ could plausibly claim that site $B$ was down or heavily overloaded for some significant period of time (causing requests to be dropped), and thus no referrals were completed (or thus credited to $A$'s account) during that time. An approach that enables $A$ to additionally record a lower bound on its number of referrals is the topic of Section 3.

[1] Other definitions are possible; e.g., the number of click-throughs for which $A$ should be paid might not include those in which $B$ responded with an error message. Nevertheless, in most cases the claims we make for our schemes continue to hold even under other reasonable definitions.

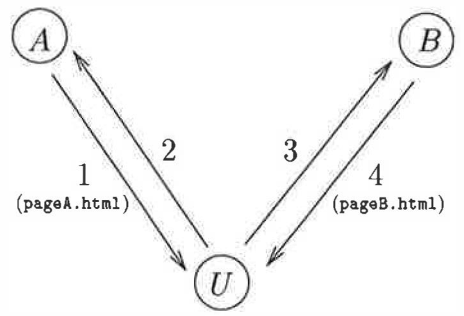


Figure 2: **Bounding referrals from above**: User $U$ retrieves **pageA.html** from $A$ (message 1). When the user clicks on the link to **pageB.html**, the user's browser sends a notification to $A$ (message 2) and then retrieves **pageB.html** (messages 3,4).

### 2.1 Using HTTP redirection

The first approach we describe for achieving the interaction of Figure 2 uses the "redirect" feature of the HTTP protocol. When a browser requests a URL from a web server, the server can return an HTTP redirection status code in the range 300–399 (e.g., **301 Moved Permanently**), which indicates to the browser that it should look for the page at another URL. This other URL is specified in the **Location** HTTP header. Upon receiving a response with status code **301 Moved Permanently**, the browser finds the **Location** header and immediately issues a request for the URL specified in that field.

Given this mechanism, one way for site $A$ to monitor the number of clicks through **pageA.html** to **pageB.html** is to craft **pageA.html** so that its link purportedly to **pageB.html** is really a link to a "dummy" URL on site $A$. If site $A$ is configured to redirect requests for this dummy URL to **pageB.html** on site $B$, then $A$ can easily monitor clicks through **pageA.html** to **pageB.html** by monitoring the number of requests for the dummy URL. The resulting web transaction proceeds as shown in Figure 2: after receiving **pageA.html**, the user clicks on the link purportedly to **pageB.html**, which causes the dummy URL on site $A$ to be requested (message 2). $A$ returns an HTTP redirect header with the **Location** field set to the URL of **pageB.html**, which causes the browser to retrieve **pageB.html** (messages 3,4). The number of requests for the dummy URL is an indicator of the number of clicks through **pageA.html** to **pageB.html**.

One practical obstacle to this approach as described so far is that it employs a reconfiguration of the web server on site $A$, which may not be possible if the participant in the click-through program does not have the authority to reconfigure the web server on site $A$. It is possible to effect this redirection without reconfiguring the web server by using CGI programming. On many web servers, a CGI script that returns a properly formatted **Location** header will cause a redirection to the URL named in that header. So, if the dummy URL on site $A$ is the URL for a CGI script that outputs a **Location** header set to the URL of **pageB.html**, then this achieves the exchange of Figure 2. This exchange can also be achieved by using a *no parse header* (NPH) script, which is a CGI script that is allowed to entirely control the HTTP headers in the response sent back to the browser. An NPH script that explicitly returns a redirection status code and **Location** header can also be used to effect the desired redirection.

It is worth noting that some obvious HTML-only approaches to effecting this redirection do not suffice because they cause the HTTP **Referrer** field to be blanked in the request to $B$, thereby precluding $A$ from getting credit for the click-through. One such approach is to craft **pageA.html** so that its link purportedly to **pageB.html** is really a link to an HTML page on site $A$ that immediately "refreshes" the user's browser to **pageB.html** using HTML's **<meta>** tag (see [MK97, Section 14.2]). Using this approach with NC4 and IE4, the **Referrer** field that $B$ received was empty.

### 2.2 Using JavaScript

In this section, we describe a second way of achieving the message exchange shown in Figure 2. The main difference of this approach from that of the previous section is that the message exchange is achieved by embedding JavaScript commands in **pageA.html**, rather than employing HTTP redirection. It is also instructive for introducing techniques that will be useful in Section 3.

Suppose that when signing up for $B$'s click-through payment program, $A$ is instructed by $B$ to place the following link to $B$ in its page:

```
<a href="http://siteB.com/pageB.html">
Click here for site B.
</a>
```

In this approach, the webmaster of $A$ constructs its **pageA.html** as follows. First, she makes a file **pageAcontents.html** that contains the (HTML commands to generate the) actual contents that she wants to display to the user, including the link to site $B$. This file looks as shown in Figure 3. The important aspect of **pageAcontents.html** is the **onClick** attribute added to the link to $B$. When the user clicks this link, the browser first executes the JavaScript code in the **onClick** attribute before retrieving **pageB.html**. In this case, the **onClick** attribute invokes a function called **notify**, defined in **pageA.html** as shown in Figure 4.

The page **pageA.html** consists of a header containing a JavaScript function **notify**, and a body consisting of two frames: one named **visible** and displaying **pageAcontents.html** (the file in Figure 3), and the other named **invisible** and initially blank. As their names suggest, the **visible** frame consumes 100% of the browser window (see the **<frameset>** tag); the **invisible** frame is truly invisible to the user. When the user clicks the link to **pageB.html** in **pageAcontents.html**, this invokes the **notify** function of **pageA.html** with the URL of **pageB.html**. The **notify** function requests a URL on site $A$. This URL serves the same purpose as the dummy URL of Section 2.1, i.e., monitoring requests for this URL is the means by which $A$ keeps track of the clicks through **pageA.html** to **pageB.html**. For example, here this URL is the URL of a CGI script on site $A$ (**record.cgi**), which is provided the URL of **pageB.html** as input (following the ?) for recording. Because JavaScript does not support general networking but does support fetching URLs, the **notify** function invokes **record.cgi** in a roundabout way, namely by fetching the output of the CGI script and "displaying" it to the user in the **invisible** frame. It does this by assigning the **location** property of the **invisible** frame to be the URL of the CGI script. When invoked, **record.cgi** simply records the referral to **pageB.html** and returns.

Using this simple trick, the click-through sequence has been transformed from that in Figure 1 to that in Figure 2. The browser invokes **record.cgi** on site $A$ (message 2) before retrieving **pageB.html** from site $B$ (messages 3,4). By using logs kept by **record.cgi**, site $A$ can monitor an upper bound on how many referrals its pages have made to $B$.

The main risk to the claim that $A$ maintains an upper bound on its referrals to $B$ with this scheme is that the browser's connection back to $A$ (message 2) might be preempted by the retrieval of **pageB.html**. This is conceivable if (i) the setup of the connection back to $A$ is delayed, e.g., due to network congestion, and (ii) **pageB.html** is a page that overtakes the top-level browser window (as opposed to displaying in the **visible** frame only), thereby overwriting **pageA.html**. If this is deemed a significant risk, then **pageB.html** can be displayed in a separate browser window (so that **pageA.html** is not overwritten) by including a **target** attribute in the link to **pageB.html** in **pageAcontents.html**.

## 3 A lower bound on referrals

In this section we describe a somewhat different approach to recording the number of referrals that $A$ gives to $B$. The method of this section addresses one limitation of those in Section 2, namely that the referral count recorded by $A$ is only an upper bound on the number of referrals it gives $B$. The solution in this section enables $A$ to infer a *lower bound*, i.e., a number of referrals for which $A$ has confidence that $B$ actually received the referred request. To achieve this, we modify our strategy so that $A$ is notified by the user's browser *only if* $B$ responds to the browser's request for **pageB.html**. That is, our goal is a protocol like that shown in Figure 5, where the browser first retrieves **pageB.html** (messages 2,3) and then informs $A$ of this afterwards (message 4).

Achieving the interaction of Figure 5 is more complicated than the simple tricks of Section 2. The general strategy that we take is as follows. When the link to **pageB.html** in **pageA.html** is clicked by the user, **pageA.html** opens a new browser window, named **nextpage**, and directs **pageB.html** to be displayed there. This enables JavaScript embedded in **pageA.html** to continue to run in the original window while **pageB.html** is being loaded. The goal then is for the **pageA.html** script to detect when the **nextpage** window has received a response from site $B$ (i.e., message 3 in Figure 5), indicating that $B$ has received the HTTP request for **pageB.html** including the **Referrer** field crediting $A$ for the refer-

```
<html>
<!-- File: pageAcontents.html -->
...
<a href="http://siteB.com/pageB.html"
   onClick="parent.notify('http://siteB.com/pageB.html')">
Click here for site B.
</a>
...
</html>
```

Figure 3: File pageAcontents.html for scheme of Section 2.2

```
<html>
<!-- File: pageA.html -->
<head>
<script language="JavaScript">
function notify(url) {
    invisible.location="http://siteA.com/cgi-bin/record.cgi?refer=" + url;
}
</script>
</head>

<frameset rows="100%,*">
<frame src="pageAcontents.html" name="visible">
<frame src="about:blank" name="invisible">
</frameset>

</html>
```

Figure 4: File pageA.html for scheme of Section 2.2

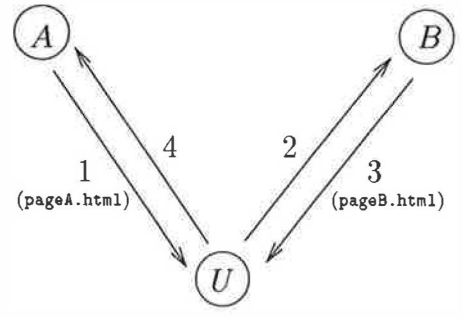


Figure 5: **Bounding referrals from below**: User $U$ retrieves pageA.html (message 1) and clicks on a link in it, causing pageB.html to be requested (message 2). Only if $B$ responds (message 3), the browser notifies $A$ of the referral (message 4).

ral. When it detects this, the script in pageA.html causes a URL on site $A$ to be requested, thereby notifying $A$ of the referral (message 4 in Figure 5).

The complexity in this approach is in the means by which the pageA.html script detects that site $B$ has responded. A first attempt might be for the script to set the onload event handler for the nextpage window when that window is created. The onload event handler is invoked when pageB.html finishes loading into the nextpage window (see [Fla98]). Thus, if pageA.html sets the onload event handler to be a function that notifies site $A$ of the referral, this would achieve the exchange of Figure 5. While this works with NC4, it does not work with IE4: presumably for security reasons, IE4 clears the nextpage window's onload event handler before loading pageB.html, and so $A$ is not notified when pageB.html has loaded. Moreover, NC4's failure to clear the onload event handler is arguably a weakness in its security model that should disappear in future versions of the browser (see [AM98]).

Fortunately, security mechanisms similar to those that cause this approach to fail for IE4 can be exploited in both browsers to achieve the effect we desire. The approach we take is for the script in pageA.html to periodically probe the JavaScript namespace of the nextpage window. Before $B$ has responded, these probes will be allowed by the browser. After $B$ has responded, however, these probes will be disallowed by the browser's security mechanisms (see [Fla98, Chapter 21]), causing a JavaScript error. By specifying an appropriate error handler for this error, the script in pageA.html can notify site $A$ of the referral. In the remainder of this section, we present an implementation of these ideas. For simplicity, our implementation here is not fully general; in particular, it suffices only for the case in which pageA.html offers a link to only one target site $B$. However, it can be generalized so that pageA.html can offer multiple target sites.

As in Section 2.2, our solution here is structured using a file pageAcontents.html to hold the actual contents of the page that $A$ wants to display to the user (including the link to $B$'s page), which is served to the user within pageA.html in a frameset. The file pageAcontents.html now looks as shown in Figure 6. The two differences from the previous pageAcontents.html (Figure 3) are the addition of a target attribute in the link and the invocation of setup (vs. notify) in the onClick event handler. Due to the latter, when the link is clicked, now the setup function is invoked, which is defined in pageA.html as shown in Figure 7.

When invoked, the setup function opens a new browser window named nextpage (as specified in the second argument of the window.open method call). This name is the value of the target attribute of the link to pageB.html in pageAcontents.html (see Figure 6), which means that pageB.html will be displayed in this new window when it is eventually retrieved. To ensure that the script in pageA.html is allowed to probe the namespace of nextpage until $B$ has responded, setup initially writes a simple HTML page into nextpage, using the document.open, write, and close methods.

The probes into the namespace of the nextpage window are performed by the probe function. The probe function attempts to read a portion of the namespace of the nextpage window (in this case, its location.href property) that will cause an error after $B$ has responded but will be allowed beforehand. If its read is allowed, then the probe function sets a timer so that it is invoked again 100 milliseconds later. Otherwise the error handler for the window in which this script is running, i.e., the window displaying pageA.html, is invoked. This error handler schedules an invocation of the notify function (see the line before the </script> tag in Figure 7).[2] As in Section 2.2, this function invokes the record.cgi CGI script on site $A$ with the URL

[2] The scheduled delay (in Figure 7, of one second) before invoking notify avoids various race conditions in IE4, yielding a more robust implementation for this platform.

```
<html>
<!-- File: pageAcontents.html -->
...
<a target="nextpage"
   href="http://siteB.com/pageB.html"
   onClick="parent.setup('http://siteB.com/pageB.html')">
Click here for site B.
</a>
...
</html>
```

Figure 6: File pageAcontents.html for scheme of Section 3

of pageB.html, which was stored in the retrieved variable during the execution of setup. Again, the trick of assigning to the location property of an invisible frame is used to invoke record.cgi.

To summarize, this achieves the mechanism shown in Figure 5: when the link to pageB.html in pageAcontents.html is clicked by the user, (i) the setup function is invoked to open a new browser window; (ii) pageB.html is retrieved and displayed in that window (messages 2,3); (iii) an error is encountered in probe; (iv) the error handler is invoked to schedule notify, which (v) invokes record.cgi on site $A$ with the URL of pageB.html (message 4). Moreover, if $B$ does not receive message 2, then neither message 3 nor message 4 will be sent. Like the mechanisms of Section 2, this technique requires no cooperation from $B$ for $A$ to track the referrals its pages give to $B$. And again, this technique presents nothing out of the ordinary to the user; spawning new windows is not uncommon while following links to other sites.

The main factor limiting the accuracy of $A$'s referral counting with this scheme appears to be the risk that the user closes the window containing pageA.html before notify is invoked. In this case, the script in pageA.html will be halted before site $A$ is notified of the referral. Thus, at best we can claim that this mechanism reports a lower bound on the number of referrals that $A$ gives to $B$. While there are other potential sources of inaccuracy, we believe that those of which we are aware can be discounted or virtually eliminated. For example, there is a risk that the user aborts the loading of pageB.html before receiving a response from site $B$ and instead loads a different page in the nextpage window, in which case the referral notification to $A$ would be erroneous. However, this risk is mitigated if the window is created with no location line or toolbar,[3] as is achieved by the third argument of the open call in Figure 7. Another risk is that some party invokes record.cgi arbitrarily, and in particular when no referral has taken place. However, there seems to be little practical motivation for such "attacks".

Because this scheme reports only a lower bound on referrals, perhaps the most prudent use of it is in combination with that of Section 2. Combined in the obvious way, these mechanisms enable site $A$ to retain both an upper and lower bound on the number of referrals that it has given to $B$, and typically these numbers should be very close to one another. A large gap in these bounds indicates to the webmaster of site $A$ that she should examine the availability of site $B$. Even without further evidence, large discrepancies between these upper and lower bounds may be good cause to no longer advertise $B$'s pages.

## 4 Cooperative approaches

Though the schemes of Sections 2 and 3 enable participants in a click-through program to approximate the number of click-throughs for which they should

[3] This does not completely prevent a user from loading a different page into the nextpage window before pageB.html loads: e.g., the user may still use "drag and drop" features or keyboard shortcuts to load a different URL into the window. However, we expect that the percentage of users employing these mechanisms is small and thus that users loading different pages into the nextpage window will yield insignificant error in the lower bound, especially since there is typically a very limited time frame (i.e., before site $B$ responds) in which the user would need to load the different page.

```
<html>
<!-- File: pageA.html -->
<head>
<script language="JavaScript">
var retrieved = null;
var w = null;

function setup(url) {
        retrieved = url;
        w = window.open("", "nextpage", "scrollbars,resizable,status");
        w.document.open("text/html");
        w.document.write("<html></html>");
        w.document.close();
        probe();
}

function probe() {
        if (w.closed) return;
        var temp = w.location.href;
        setTimeout("probe()", 100);
}

function notify() {
        if (w.closed) return;
        invisible.location="http://siteA.com/cgi-bin/record.cgi?refer=" + retrieved;
}

window.onerror = function() { setTimeout("notify()", 1000); return true; };
</script>
</head>

<frameset rows="100%,*">
<frame src="pageAcontents.html" name="visible">
<frame src="about:blank" name="invisible">
</frameset>

</html>
```

Figure 7: File **pageA.html** for scheme of Section 3

be paid, there is still some room for error in these approaches. In this section, we show that even greater accuracy can be achieved if site $B$ cooperates with site $A$ to enable $A$ to more effectively monitor $B$'s behavior. While the schemes of this section require cooperation by site $B$, this does not imply that $A$ must fully trust $B$. Rather, if $B$ misbehaves, then it risks detection by site $A$, with high probability if $A$ combines the approaches of this section with those of Sections 2 and 3. Site $B$ might be willing to cooperate in these schemes to instill trust in its referrers, in the hopes of obtaining more clients for its click-through program.

### 4.1 Click-through acknowledgements

In the first approach that we propose, site $B$ effectively "acknowledges" each referral from $A$ as the click-through happens. $B$ could send an acknowledgement to $A$ directly, i.e., by sending it in a message to $A$, but this requires $B$ to incur more costs (e.g., connection setups) than is necessary. Rather, here we review a simple way in which $B$ can piggyback the acknowledgement on its reply to the user, so that the user's browser will forward the acknowledgement to $A$.

Transferring an acknowledgement from $B$ to $A$ via the user's browser can be achieved easily with the addition of a CGI script to site $B$ and some modifications to `pageB.html`. To begin with, $B$ sets up a CGI script that serves $B$'s web pages (possibly only for referrals from click-through program participants). Let's call this script `siteB.com/cgi-bin/serve.cgi`. This CGI script accepts as input the name of a page to produce (e.g., `pageB.html`) and emits a version of `pageB.html` that is slightly different for each referred request. Specifically, if $B$ receives a request for `pageB.html` referred by $A$, then the version of `pageB.html` served by `serve.cgi` requests a dummy URL on site $A$ when it loads. Each retrieval of this dummy URL is an implicit "acknowledgement" from $B$.

A more explicit acknowledgement can be achieved if the dummy URL on site $A$ is a CGI script that `pageB.html` can invoke with $B$'s site name and the time of the referral. For example, `pageB.html` emitted from `serve.cgi` might look as shown in Figure 8. Notice that the visible contents of the page, `pageBcontents.html`, are served within a `frameset` with one visible frame and one invisible frame (analogous to how $A$ served `pageA.html` in Sections 2.2 and 3). As the `frameset` loads, the browser invokes $A$'s `record.cgi` with the arguments provided by $B$. Again, the trick of invoking `record.cgi` by writing its output to an invisible frame is used, but this time it is done by `pageB.html` (vs. `pageA.html`). Alternatively, `record.cgi` could be invoked from an `<img>` tag, for example. The `record.cgi` CGI script on site $A$, upon being invoked, can verify that the arguments properly acknowledge $A$'s referral.

Because $B$ could serve a `pageB.html` that does not invoke $A$'s `record.cgi`, it is advisable for $A$ to construct `pageA.html` as in Section 2, i.e., so that $A$ is informed whenever the user clicks on the link to `pageB.html`. This will alert $A$ if $B$ routinely serves a `pageB.html` that does not invoke the appropriate callback to $A$'s `record.cgi`. $A$ could further employ the mechanism of Section 3, providing $A$ with the full detection capabilities offered by both approaches. In this light, the technique of this section can be viewed as a way for $B$ to help $A$ improve its click-through monitoring over what $A$ can achieve without $B$'s help using the schemes of Section 2 and Section 3. In particular, $B$'s acknowledgement may reach $A$ even if the notification from the scheme of Section 3 does not (e.g., because the user prematurely closes the window containing `pageA.html`). If $A$ couples the detection techniques of Sections 2 and 3 with random inspections of `pageB.html` as served by $B$ on a referral, $B$ stands a high probability of being caught if it fails to acknowledge $A$ a significant portion of the time.

### 4.2 Click-through nonrepudiation

One drawback of all our previous schemes is that while they enable a referrer to detect hit shaving, they do not arm the referrer with any evidence to present to a third party in the case of a dispute. So, in the extreme, the webmaster of site $B$ can repudiate some or all referrals, including any acknowledgements it sent, and refuse to pay certain referrers. While these referrers are thus likely to leave $B$'s click-through program, there is nothing that they can do to bring third-party leverage on the dispute. In this section we extend the technique in Section 4.1 to enable $B$ to pass nonrepudiable acknowledgements to the referrer. Again, any failure of $B$ to cooperate is quickly detectable by the referring site $A$ (if combined with the techniques of

```
<html>
<!-- pageB.html, dynamically generated by serve.cgi -->

<frameset rows="100%,*">
<frame src="pageBcontents.html">
<frame src="http://siteA.com/cgi-bin/record.cgi?refer=siteB.com&when=Feb2_13:04_EST_1998">
</frameset>
</html>
```

Figure 8: File **pageB.html** in scheme of Section 4.1

Sections 2 and 3), and so the webmaster of $A$ can immediately take action to avert a dispute, rather than wait until, say, the end of the month to find out that $B$ will not pay her.

### 4.2.1 Using digital signatures

If there is a well-known public key for authenticating site $B$ via digital signatures (e.g., [RSA78]), then one approach for $B$ to provide nonrepudiable acknowledgements to $A$ is for $B$ to pass a digital signature to $A$ as part of the click-through protocol. This signature could sign a tuple containing the IP address of the user, the time and date of the referral, the page to which the referral was made, and the referring page. $A$ can then retain this signed tuple for use in a dispute with $B$ later, if necessary. Like in Section 4.1, $B$ can create this signature in **serve.cgi** and include it within **pageB.html**, to be passed as an argument to a CGI script on site $A$ by the user's browser when **pageB.html** loads.

A drawback of this approach is that it requires $B$ to compute a digital signature per referral, which must be done on its critical path for servicing the user's request. Because digital signatures, particularly RSA signatures [RSA78], tend to be computationally intensive, the additional computational load imposed by these signatures may be prohibitive if $B$ is a very busy server.

### 4.2.2 Using hash chains

In order to lessen the computational burden on $B$, in this section we sketch an approach that requires far less from $B$ computationally but that still provides some degree of nonrepudiable evidence to $A$. It employs the well-known idea of *hash chaining*, which has been used in the past for efficient user authentication [Hal94] and micropayments [RS95], among other things.

Again we assume that there is a well-known (i.e., authenticated) public key for $B$. When $A$ signs up for $B$'s click-through payment program, $B$ generates a large, unpredictable number $s$, applies a one-way hash function (e.g., [SHA95]) $f$ to it $k$ times to produce $\ell = f^k(s)$, digitally signs the pair $<k, \ell>$, and sends $<k, \ell>$ and the signature to $A$. Note that all of this takes place when $A$ registers for the click-through program, not on the critical path of referrals. Once this is set up, rather than passing a digital signature back to $A$ during a referral, $B$ simply passes back the pair $<i, \ell'>$ to $A$, where $\ell' = f^{k-i}(s)$, for the $i$-th referral ($1 \leq i \leq k$) that $A$ gives it. $B$ can pass the pair $<i, \ell'>$ to $A$ using techniques like those of Section 4.1. $A$ can verify the correctness of this pair by verifying that $\ell = f^i(\ell')$. In the event of a dispute, $A$ need only present the digitally signed pair $<k, \ell>$ and a pair $<i, \ell'>$ where $\ell = f^i(\ell')$ to convince a third party that $B$ received at least $i$ referrals from $A$.

This scheme has some disadvantages in comparison to that of Section 4.2.1. The main disadvantage is that $B$ can later repudiate the user's IP address in this scheme. In payment programs that pay only for "unique" referrals, $A$'s inability to record the user IP address in a way that prevents $B$ from later repudiating it could leave $A$ at a disadvantage in a dispute. An agreement that $B$ returns a referral record (i.e., a new pair $<i, f^{k-i}(s)>$) only for unique referrals may restore the balance, but only if $A$ is prepared to verify, when $B$ refuses to return a referral record, that the referred user was a repeat user. A second disadvantage of this scheme is that it must periodically be "refreshed"; i.e., once $k$ referrals from $A$ to $B$ have been made, then $A$ must obtain a new signed pair $<k', f^{k'}(s')>$ from $B$.

## 5 Discussion

As mentioned in Section 4.2.2, our use of hash-chaining is similar to its use in certain micropayment schemes, specifically the PayWord scheme due to Rivest and Shamir [RS95]. This similarity is perhaps not coincidental, in that the deployment of a micropayment scheme, or more generally any digital cash scheme, could be a useful tool to counter hit shaving. In this case, the target of a referral could be required to pass a digital coin back to the referrer in the referral protocol, e.g., using the techniques of Section 4. The referrer could thus collect immediate payment for referrals it gives, and detect when payment is not being received.

Other potential developments that could expand our options for countering hit shaving include the adoption of a richer security model for JavaScript. For example, Anupam and Mayer [AM98] propose a JavaScript security model in which a script can selectively allow other scripts to access portions of its namespace by configuring access control lists accordingly. If adopted, this could enable other cooperative solutions in which **pageB.html** allows a script in **pageA.html** to access some portion of its namespace, so that the script in **pageA.html** can confirm when **pageB.html** has loaded and notify site $A$. Such solutions have the advantage of allowing **pageB.html** to be a static page (as opposed to one dynamically generated by a CGI script on site $B$), though they also place requirements on **pageA.html** that the solutions of Section 4 do not.

Although the techniques proposed in this paper are effective for detecting hit shaving, they do have the adverse effect of eroding user privacy further than the web already does today. That is, the web today, via the **Referrer** HTTP header, often reveals to a site the page that a user visited previously. Our techniques further enable the referring site to learn the page that the user visits next. Mechanisms for anonymously surfing the web, such as the Anonymizer, the Lucent Personalized Web Assistant [GGMM97], and Crowds [RR98][4] are generally incompatible with click-through payment programs on two counts: they strip out the **Referrer** field, and they preclude monitoring of user IP addresses for the purposes of detecting hit inflation. The former can be remedied by configuring these systems to let the **Referrer** field remain; the latter obstacle appears more difficult to overcome.

This work leaves several open problems. In particular, we have not attempted to address the problem of hit inflation, but have only attempted to not exacerbate it. More robust approaches for detecting or preventing hit shaving should also be explored.

**Acknowledgments** We are grateful to Avi Rubin of AT&T Labs and the anonymous referees for providing encouragement and helpful comments.

## References

[AM98] V. Anupam and A. Mayer. Security of web browser scripting languages: Vulnerabilities, attacks, and remedies. In *Proceedings of the 7th USENIX Security Symposium*, January 1998.

[Fla98] D. Flanagan. *JavaScript: The Definitive Guide*. 3rd edition, O'Reilly & Associates, 1998.

[GGMM97] E. Gabber, P. Gibbons, Y. Matias, and A. Mayer. How to make personalized web browsing simple, secure, and anonymous. In *Proceedings of Financial Cryptography '97*, 1997.

[Hal94] N. M. Haller. The S/Key$^{\text{TM}}$ one-time password system. In *Proceedings of the Internet Society Symposium on Network and Distributed Systems*, 1994.

[Kle98] D. Klein. *Succumbing to the dark side of the force: The Internet as seen from an adult web site*. Invited talk at the 1998 USENIX Annual Technical Conference, June 17, 1998.

[MK97] C. Musciano and B. Kennedy. *HTML: The Definitive Guide*. 2nd Edition, O'Reilly & Associates, 1997.

[RR98] M. K. Reiter and A. D. Rubin. Crowds: Anonymous web transactions. *ACM Transactions on Information and System Security* 1(1), June 1998.

[RS95] R. Rivest and A. Shamir. PayWord and MicroMint: Two simple micropayment schemes. Manuscript, 1995.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21(2):120–126, February 1978.

[SHA95] FIPS 180-1, Secure hash standard. Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, April 17, 1995.

[4] See `www.anonymizer.com`, `lpwa.com`, and `www.research.att.com/projects/crowds`, respectively.

# Sales Promotions on the Internet

Manoj Kumar, Anand Rangachari, Anant Jhingran and Rakesh Mohan
{mkumar, anand, anant, rakesh}@watson.ibm.com
*IBM Research Division*
*T.J. Watson Research Center*
*Yorktown Heights, NY 10598*

## Abstract

*We describe a sales promotion application for distributing and redeeming coupons on the Internet during online shopping. Various types of sales promotions and coupons, and the methods used to target coupons to select potential buyers are reviewed. Security mechanisms needed to prevent alterations, duplication, and trading of coupons by customers, and fraudulent use of manufacturer's coupons by retailers are identified. An implementation of electronic coupons is described. The impact of coupon trading and duplication, facilitated by the Internet, on the effectiveness of sales promotion campaigns is discussed.*

## 1. Introduction

Sales promotions are important marketing tools in today's businesses. They command a greater portion of the marketing budget than advertisements [1] (in consumer-packaged-goods business). However, while advertisements are quite popular and a big business on the Internet, sales promotions on the Internet have not caught on yet. Part of the reason is that an advertisement is purely informational with no exchange value. It is broadcast to the largest possible segment of population possible within a budget. On the other hand, coupons, the primary vehicle for sales promotion, have an exchange value and are intended for a select section of the population. The digital nature (ability to make perfect copies inexpensively), easy access, and low overhead for distributing information on the Internet is a boon to advertising, but a problem for sales promotions.

In this paper we discuss the use of the Internet for distributing coupons. We describe the software needed to issue and redeem coupons on the Internet. The three attributes of the Internet we focus on are: 1) the ability to monitor the user's online shopping behavior (click stream analysis) to issue coupons; 2) mechanisms for embedding coupons in advertisements and publications and potential for fraud due to uncontrolled duplication and distribution of coupons; and 3) ease in capturing the coupons and redeeming them which will result in e-coupons being used by buyers who previously did not have sufficient incentive to cut the coupon and remember to carry it to the store.

We begin this paper by discussing different types of paper coupons in use today and draw the distinction between manufacturer's coupons and store coupons. Store coupons are issued by a merchant to attract shoppers in his local area to his store. Store coupons are primarily used to: 1) manage (reduce) inventory; 2) reward customer loyalty; and 3) attract buyers to the physical store by discounting select few products. Manufacturer's coupons are typically issued by manufactures of national brands. They are used to: 1) gain market share by switching buyers from competing brands, especially for newly introduced brands; 2) match competitors coupon campaign; 3) and increase repurchase rates among occasional users of a brand. *The common theme linking all of the above business objectives is the requirement to reduce prices temporarily.* Most marketing text books have extensive coverage of various sales promotions methods, [1] being one of them.

Next we focus on preventing the duplication and trading of coupons. The Internet allows buyers possessing digital coupons to duplicate them arbitrarily many times, and to distribute the copies effortlessly around the globe. This not only defeats the manufacturer's objective of targeting the coupons, but also renders him incapable of estimating how many coupons will be redeemed and thus preventing him from budgeting for a coupon campaign. Retailers can also get hold of one manufacturer coupon, make unlimited copies of them, and either pass them to buyers not targeted by the manufacturer (to improve their sales and profits), or worse, redeem the coupon on behalf of the buyers and pocket the difference. We discuss methods to prevent and control such fraud.

Then we describe the software required to issue and redeem coupons on the Internet. In section 5 we describe the components of an electronic coupon. Then in section 6 we describe the implementation of a store coupon application. We have prototyped the mechanisms to display coupons to the buyers, and the mechanisms to store coupons and redeem them at the time of purchase. In the last section we discuss the impact of duplication and trading of electronic coupons (facilitated greatly by the Internet) on the effectiveness of a sales promotion campaign.

Electronic coupons are already offered on the Internet. In most cases, the coupon is an image that can be printed to created a paper coupon which is then taken to the store. In some cases, only a bar code is printed at the shopper's terminal which is scanned at a kiosk at the store to print the corresponding coupon. Finally, at cash registers in many grocery stores, the purchase order is analyzed at checkout to determine the coupons to be offered for future shopping visits, and these coupons are printed at the cash register. Third party intermediaries are also emerging as coupon distributors. They promote a site where shoppers can come and collect coupons from various manufacturers and stores.

## 2. Different types of traditional coupons and e-coupons

There are many different kinds of coupons. A *typical coupon* is a piece of paper or electronic document, which is either distributed widely in a certain geographic area (mass mailing), or mailed to selected shoppers in a area (direct mailing). In the next section we will discuss how individuals are chosen in a geographic area for sending coupons. These coupons typically have an expiry date of a few weeks to a few months associated with them.

*Loyalty awards* are essentially points given on purchase of some thing, which can later be redeemed for some merchandise. Frequent flyer miles is a good example. Coupons are used to handout loyalty points which can later be redeemed for merchandise. Coupons can also be used to carry out '*two for the price for one'* deals. These deals are partly loyalty award and partly an incentive to increase consumption of the product being promoted.

*Gift certificates* can also be viewed in the framework of coupons. Gift certificate is essentially a coupon issued by a store which has minimal restrictions on what merchandise can be purchased at that store, and is paid for by a shopper wishing to buy a gift rather than from the stores marketing budget. It would use much of the same technology to control duplication and fraud as regular store coupons.

Electronic version of the above mentioned type of coupons can be implemented for use on the Internet. Additionally, a new type of coupon, which has no analog in the paper coupon world and can be offered only on the Internet, is the *instant rebate*. This coupon is issued to a shopper who is vacillating about a purchase. The purpose of the coupon is to induce the shopper to buy the merchandise immediately. For example, an item can be discounted for a short time if the buyer stays on the page for a long time or returns to it often. This kind of coupon is usually good for a few minutes or hours. In this mode the use of a e-coupons is similar to price negotiation or haggling between a buyer and a seller in real world situations.

Instant rebate coupons can also be use to mark down the price of a commodity for a short period of time to create a 'daily-special' or fifteen minute special. In the latter case the instant rebate coupons will be shown indiscriminately to all shoppers rather than only the shoppers who have shown vacillation This is very similar to the price mark downs on TV home shopping channels. This differs from the haggling mentioned in the previous paragraph. Here the discounting is not based on direct observation of a buyers reluctance to buy.

As mentioned earlier, traditional coupons can be classified as *store coupons* and *manufacturer's coupons*. E-coupons will also come in both varieties. Store coupons entail a two or three party transaction as shown in Figure 1. The store either prints the coupons and directly distributes it to the buyers, or has the coupons printed in a local magazine. Buyers clip the coupon from the printed material received from the store or from the magazine and redeem the coupon at the issuing store. Manufacturers coupons are distributed to buyers either directly through mail or as inserts in products, or through publications in national

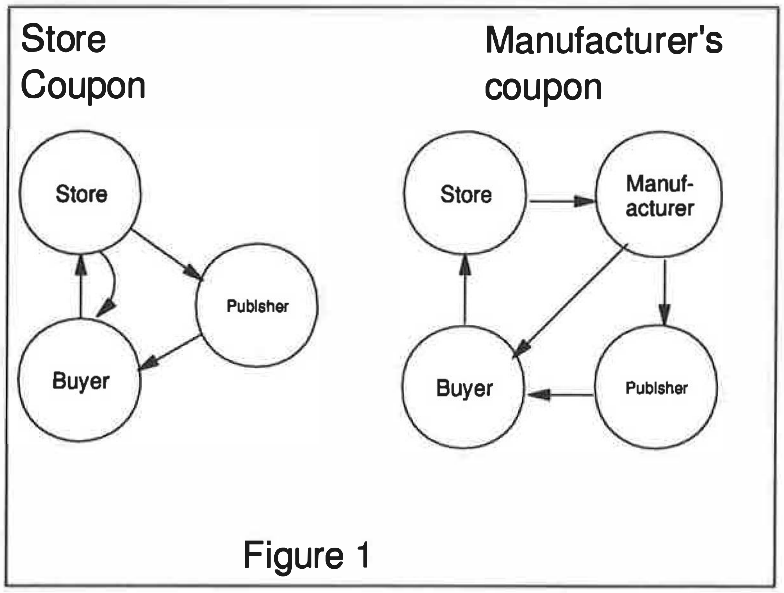


Figure 1

or regional magazines. Buyers redeem the coupons at the store and the store in turn is reimbursed by the manufacturer for the coupons it accepts.

## 3. Distributing coupons to select shoppers

Currently store coupons are generally mailed indiscriminately to local geographic area (mass mailing). They are also targeted to buyers with specific buying pattern or past purchase history at the store (direct mailing). For example, coupons for charcoal may be sent to people who bought a grill at the store, or charcoal at the store in the recent past. Manufacturer's coupons are invariably targeted either geographically or guided by buyer segmentation (by income, profession, etc.). Choosing where to publish the coupons (if they are not mailed directly) helps create this segmentation. Manufacturer's coupons are also dispensed at the point of sale, attached to the merchandise (for cross sales, up sales and immediate discounts), or presented with the product as mail in rebate.

Mechanisms to estimate a buyer's or buyer group's inclination to buy various products based on his purchase history are numerous and well known and currently used to target advertising and coupons. Cross selling and up selling are other important uses of traditional coupons. In cross selling, the purchase of some product, say dress shirt, creates the knowledge that the buyer is possibly also well disposed to buying ties, and he can be encouraged to buy ties at a later date by sending him a coupon for a tie. In upselling incentives are given at the time of purchase of a product to buy upgrades or accessories to that product. For example, the purchase of a personal computer and not a printer may suggest that the coupon for discount on printers should be sent to the buyer.

The Internet improves the effectiveness of targeting coupons to buyers who would have otherwise not bought the product being discounted. An essential component for effective targeting is the software to make decisions on which coupons (if any) to show to a particular buyer and when. Buyers should be given coupons for products that are just below their buying threshold. The up sales coupons and cross sales also can be made more effective by using instant rebates instead of the traditional mailed coupons.

Maintaining a profile of buyers interests, likes and dislikes is another way to determine which products fall just under each buyer's buying threshold. These profiles will be populated by: 1) summarization and analysis of buyer's past purchasing behavior, 2) information obtained from external sources such as county records, department of motor vehicles, local schools etc., and 3) Information provided voluntarily by the buyer through interactive dialogues, sweepstakes, and games. The external sources mentioned earlier tell a seller what kind of car does the buyer drive, how many children he has and of what ages, what type of house does he live in. This can be used to formulate rules for what products should be promoted to a buyer.

Click stream analysis allows a merchant to instantaneously gauge the interest of a web site visitor on different products and product categories. He can then use this information to generate instant rebate coupons on the fly. This can not be done with paper coupons. For example, if a buyer scans several brands of some product and then switches to scanning a different product without selecting the first one for purchase, it may be an opportunity to give a buying incentive for the first product through an e-coupon. Similarly, if the buyer revisits a high priced product several times without buying it, or spends a long time on details of a product, the product may be ripe for instant rebate. Click stream analysis coupled with information from the buyers profile will make couponing on the Internet extremely effective.

It is important that the rules for issuing coupons and instant rebates not be obvious to the buyers. Otherwise the buyers can simply exhibit the appropriate behavior to get the rebates even if they would have otherwise bought the product at its regular price. This situation is similar to the current situation of buyers waiting for after Christmas sales.

Based on the preceding discussion we can classify coupons by two attributes, *targeting* and *distribution*, as shown in Figure 2. The targeting attribute is a measure of sophistication used in selecting the buyers who will receive the coupons. The distribution attribute is a measure of control exercised in restricting the number of coupons distributed. As we will see in the next section, the security features needed to implement coupons are dictated by the need to preserve these two attributes.

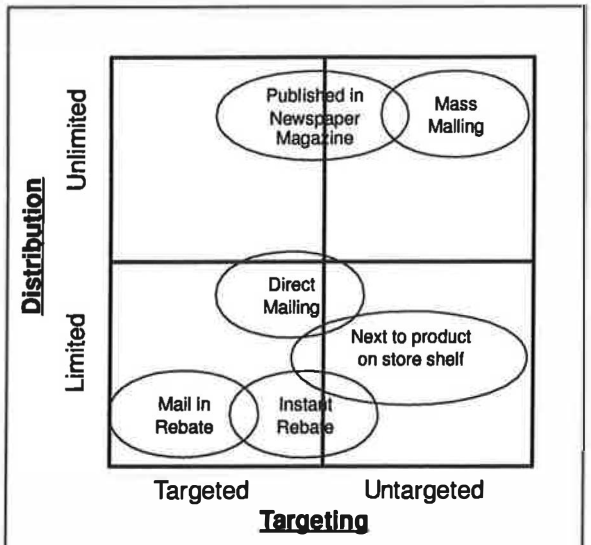


Figure 2

To simplify discussion we will deal with four types of coupons, untargeted and unlimited distribution, untargeted and limited distribution, targeted and unlimited distribution, targeted and limited distribution, corresponding to the for boxes in Figure 2. The distinction between adjacent boxes is fuzzy. We can have a mixture of the two. Targeting can be based on geography, demographics, etc., and we can be selective about the dimensions on which the coupon is targeted. In other words targeting can be for very large groups (nations or continents), or very small groups down to the level of individual. Targeting alludes to the manufacturers desire to choose who gets the coupons.

Distribution is never truly unlimited because it is limited by the distribution mechanism itself. Even with mass mailing one can only access people who can be reached by bulk rate US postage. Here we are basically alluding to the manufacturer's desire to limit the number of coupons distributed regardless of whom the coupon is distributed to.

## 4. Security issues in E-Coupons

Various digital cash schemes provide a good starting point to control fraud in e-coupons. However, duplicate spending by the digital cash holder can be prevented only by online verification, an expensive proposition for manufacturer coupons of small denominations. If online verification is not used, the bank can at best determine whether the buyer carried out the duplication or the seller did it [2]. Anonymity is often not a big concern in e-coupons, and hence the known digital cash algorithms can be simplified. Finally, digital cash algorithms assume that if the attempt to defraud is detected with sufficiently high probability, though not with certainty, sufficiently stiff penalties can be used to deter fraud. Though applicable in the world of finance, this may not hold true in the world of e-coupons. It will be difficult for Proctor and Gamble to prosecute grandma for reusing an already used coupon. In this section we discuss the security measures required to preserve the targeting and limited distribution of coupons.

### Capturing coupon on the client side:

In the electronic world, coupons will be published in a publicly accessible magazine on the Internet or on a web site. If the coupons are embedded as a gif file, plugins will be required for the browser to capture the coupon. The plug in could print the coupon directly on a printer, requiring the buyer to take the paper coupon to the store. However, the preferable method would be to capture the coupon on the buyer's system using client-side coupon application called a coupon wallet. The coupon wallet would allow the buyers to organize and search for the electronic coupons or e-coupons. When shopping in an electronic store on the Internet, the e-coupons can be redeemed directly from the coupon wallet. However, if they have to be redeemed at a physical store, they will have to be transported on a physical medium such as smart card.

## Digitally signing the coupon:

At the minimum, every manufacturer's coupon will be digitally signed by the issuer. This is needed to ensure that the three key fields of a coupon, 1) the product, or product group, to which it applies, 2) coupon value; and 3) the expiry dates are not tampered with by either the consumer or the retailer. This is also needed to prevent pranksters from introducing spurious coupons. For the latter reason store coupons must also be signed.

The above would suffice for unlimited distribution coupons. Such coupons can simply be downloaded from the site where they are offered. However, for limited distribution untargeted coupons, each coupon will also include a unique serial number, and for targeted coupons the digital identity of the coupon recipient. The digital identity could be recipients name and address, or some other equivalent identification such as digital certificate or e-mail address.

Both the serial number and digital identity, if present, will also be signed along with the item, value and expiry date fields. Now the coupon can no longer be simply downloaded from the publishers web site. A coupon-server web-application has to be invoked either at the publishers site, or the coupon issuer's site, to assign the unique serial number and incorporate the recipient's identity. Identity of the recipient could be provided to the coupon-server application by the client coupon-wallet application.

## Preventing unauthorized coupon duplication:

When coupons are valid only at a single store, the coupon sequence number solves the problem of unauthorized duplication of limited distribution coupons. The store will simply maintain a list of all redeemed coupons and thus a duplicate coupon will not be honored. If store-issued coupons can be redeemed at multiple branches of the store the list of redeemed coupons will have to be kept at a centralized redemption server serving all stores or be replicated at redemption servers distributed across stores. This approach of requiring each coupon to be screened for potential duplication would also work for manufacturers coupons. However, it is more expensive because third party Internet service providers have to be used to handle the coupon validation traffic over a large geographic area. For manufacturer coupons this approach can be used for coupons of large value, several dollars at today's Internet transaction costs.

An alternative way to avoid duplication of manufacturer's coupons is to restrict the redemption of the coupon to a particular store when the coupon is issued, and require the store to maintain the list of redeemed coupons and screen coupons for duplication before redeeming them. When a buyer wishes to receive a coupon, he is asked about which store he will go to redeem the coupon, and the store's address field is included in the coupon before the coupon issuer signs it. This assumption appears to be reasonable and is advocated also in the Millicent protocol [3].

## Preventing unauthorized coupon exchanges:

A 10% off coupon on a big appliance can be of substantial value. Unless coupon trading is curbed, coupon exchanges will surely emerge on the Internet. In the real world it is difficult for a shopper planning a purchase to locate another person who has a coupon for that purchase but does not intend to use it. Even if the person with a spare coupon can be located, the exchange of coupons is inconvenient and time consuming. Internet makes the search for spare coupon much easier, and they can be exchanged simply by e-mail. Chat rooms and coupon trading sites will surely emerge to trade coupons.

To prevent undesirable exchanging of coupons, especially the limited targeted coupons, each coupon is stamped with the digital identity of the buyer or buyer group to whom the coupon is issued, as mentioned in a previous section. This information should be verifiable at the store at the time of coupon redemption. This is easy for coupons of large monetary value which are used for large purchases such as TV's and big appliances. Stores routinely ask for this information at the time of purchase of large monetary value such a large appliances, and this information can be compared with the information contained in the coupon.

**Retailer fraud:**

The most obvious form of retailer fraud is for retailers to collect a large number of coupons from a web site that is publishing the coupons. Of course, the retailers would have to obtain those coupons through a phony set of names and addresses. Then these coupons can be redeemed by the retailer to either reward his customers or he can somehow redeem these captured coupons and pocket the proceeds. This type of fraud can be controlled by ensuring that obtaining large number of phony identities is difficult.

A practical way to handle this situation currently is to require that coupons be published at sites that require users to register before they get access to the site. News media and magazines and Internet service providers (ISP) will meet this criterion. When a buyer clicks on a coupon to capture it, the publishing site will provide the registration information about the user such as his name, address and telephone number to the coupon server. This information will be encoded in the issued coupon, as is done to prevent coupon trading. Now to capture a large number of coupons from the coupon site the retailer will have to keep either multiple subscriptions to newspapers/magazines or maintain multiple accounts with an ISP, both of which are economically unattractive. News media and ISPs would probably be willing to provide this service because it makes couponing on the Internet practical and provides coupon publishing revenue to them.

A second kind of retailer fraud is possible where a retailer makes multiple copies of a coupon he receives from some shopper. Even if retailers are required to maintain a record of the coupons redeemed at their store and prevent duplicate redemption, they can exchange lists of coupons with other retailers. In both cases the retailers will blame the duplication on shoppers. We are looking into off-line digital cash mechanisms and other approaches for solutions to this problem.

## 5. Structure of E-Coupons

An electronic coupon will contain the description of the product/package to which the coupon applies, its value, restriction on its use, and several other fields which are described next.

**The description of the product or package:**

Figure 3 illustrates the implementation of product/package and coupon value. Package is essentially a list of references to products (particular brand name or model number), or product categories (all Sony TVs, all 25" TVs, or all electronic appliances). For each item in the list, an amount is specified. Amount can be count, weight, or value of purchase. Package specifies the list of products and their quantities that a shopper is required to have in order for the coupon to be applicable. Thus, a package can specify multiple items of a kind, like 5 shirts, or bundle of different products like combination of shirts and ties. Information referenced through the product field in the package might include a text description of the product, a UPC bar code for the product or package, or a store's SKU number (for store coupons).

**Coupon value:**

The value of a coupon, as illustrated in Figure 3, is specified as a combination of one or more of the following components:

1. Discount on items in the package. Discount can be specified for all items, some items, or even no items if not both of the next two value components are null.
2. Discount on the items in the shopping basket (purchase order), which are not part of the coupon package. These are represented as optional purchases table in Figure 3.

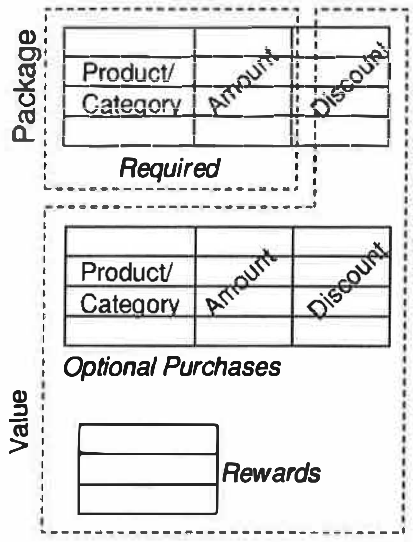


Figure 3

3. Rewards other than discount on products being purchased such as bonus points or frequent flyer miles, free fitting on clothing, or free tickets to a concert or game, or coupons for a future purchase.

The discount for required and optional purchases can be a fixed dollar amount, or a percentage of the price of the product to which the coupon is applicable. The discount could also be a function of the amount of purchase.

For simple coupons this list will have only one row in the package list and no optional purchases or rewards. Same is true for "buy one get one free" or "buy *n* items get *m* free items" type of coupons. The discount function can handle such coupons. Coupons such as "50% off on the price of shirt when you buy a pant", or "shirt free with pants" can be handled by having two rows in the package list. Coupons like "Take 20% off on up to 5 ties when you buy a suit" is handled by putting ties in the optional purchasè table. The discounts can be made much more complex by employing all the three value components with multiple entries in each of the corresponding lists (if the objective is to confuse the buyer).

**Restrictions:**

Restrictions specify the expiry dates of coupons and geographic areas where the coupons are valid or not valid. Store coupons may also include specify days of the week or time periods during a day in which coupons will be valid or not valid.

**Display support:**

Coupons are displayed to buyers at several occasions:

1. Offer to the buyer to capture the coupon.
2. Coupon displayed when buyer initiates a search or scans his coupon wallet.
3. To remind the buyer of a coupon in his wallet at an opportune time, i.e., to lock in a sale, or to create a cross sale.
4. At the time of redemption.

In an object oriented implementation, a coupon object would provide multiple display methods to handle the different display situations. Different graphics could be used with these different display methods. An elaborate display will be used to present the coupon at the publishers site to attract potential buyers' attention and encourage them to capture the coupon. This display method is part of the coupon template discussed in the next section, but not a part of each coupon. The rest would be simpler by comparison. Simplicity is required here for an added reason, to keep the size of the coupon files stored at the client side small. Information can be added in the coupon to specify the conditions under which the coupon wallet will generate reminders.

**Coupon serial number and buyer and store identity:**

The purpose of these fields was discussed earlier in the section on security issues.

## 6. Software for store-coupons

In this section we describe the software for handling store coupons. A complete store coupon application consists of the following key steps:

- Targeting the coupon to a segment of potential customers.
- Delivering the coupon to the targeted potential customers
- Providing the mechanisms for the customers to store and search for coupons
- Helping users redeem coupons that are applicable to their purchases

Figure 4 shows the steps of coupon cycle from the decision by store to issue the coupon to its redemption at the store. A store coupon application would work in conjunction with any of the commonly available Internet storefront products such as IBM's Net.Commerce. These products use relational databases that can be extended to support couponing functions. The first such extension required is to create a coupon wallet for each shopper which contains all the coupons issued to the shoppers. Wallets for all shoppers are maintained in the store application. In our current implementations, buyers do not have a coupon wallet on their personal computer to capture the coupons published by the store in a magazine.

The coupons in the coupon wallet have a link to product's description. Coupons redeemed by the shopper are retained in this file for a certain period specified by the store, but are checked off to prevent repeat redemption. This allows the store to analyze shoppers' coupon redemption behavior. The different steps of the electronic coupon application are described next.

### Seller's coupon management system:

Tools are provided for the store manager to create coupon templates. To issue coupons to buyers, the store manager selects one of the several lists or groups of shoppers and a coupon template. A coupon created from that template is generated for each shopper. Shoppers can be notified by mail, or via other online mechanisms that they have received a coupon. Coupons are also acquired by the shopper directly from various pages of the store's web site, or on buying selected products (cross-sales and up-sales). Tools are also provided to create daily specials, instant rebate, and cross-sales and up-sales coupons.

The coupon template contains information common to all coupons issued in a campaign and generates analysis data to observe the success/progress of the campaign. That would include all information specified in the previous section except the serial number, and identification of buyer and the store where the coupon will be redeemed. In addition to this information the template would specify the maximum number of coupons to be distributed, the number of coupons actually distributed and the number of coupons redeemed. Each of these numbers could be further broken down by time periods and geographic areas. As mentioned earlier, the template also has a high quality display of the coupon.

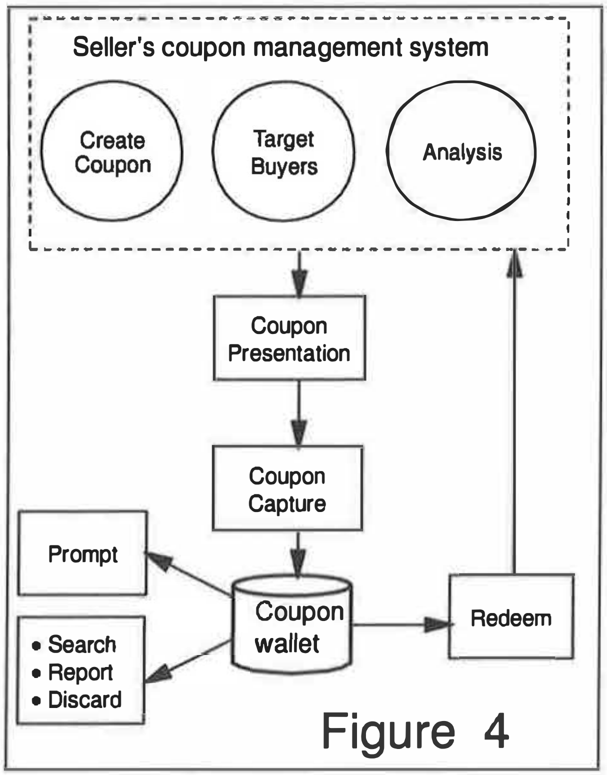


Figure 4

### Coupon presentation and capture:

Once the seller decides to issue a coupon to a set of buyers, he can place the coupon directly in the buyers' coupon wallet. The buyers are also allowed to set up filters to restrict the kind of coupons (products and brands) they receive in their coupon files. However, a preferred alternative would be to present the coupons to a buyer when he first connects to the seller's site and let the buyer select which coupons he wants to accept in his wallet. Filters can still be used to restrict the coupons that are presented to the buyer.

### Coupon search and report:

When a buyer is viewing an item in a catalog, for which he has a coupon in his wallet, he can be prompted with this coupon. He will also be prompted with instant rebate coupons which are maintained at the seller's site, rather than in his wallet.

### Coupon search and report:

Buyers can create a preference profile to indicate preferred products and product categories. Coupons displayed to a user are ordered by this preference. Search metaphors are provided to allow the buyers to search for coupons in the coupon file, and from these coupons hot links are provided to the detailed description of the product featured in the coupon. Buyers can be presented with a report on how much money they saved with coupons over some preceding time interval. This report may be requested by the buyer or presented to him voluntarily at some opportune time. Finally, when viewing available coupons, the buyers can delete the ones they no longer care for.

### Coupon redemption:

At the time of finalizing their purchase, users are presented with the applicable coupons in their coupon file and given an opportunity to redeem the coupons they choose. Since many coupons may be applicable to the same purchase, when a coupon is selected for redemption, several other coupons will become inapplicable and will be removed from display. An alternative would be select the set of coupons which results in maximum reward for the user and give the user the opportunity to accept or modify this selection.

Cross-sales and up-sales coupons are also presented to the shopper when purchases are being finalized. Finally, a user could select a coupon for which he does not have the complete set of items specified in the optional purchase part of coupon value. In this case he can be reminded about the discounts available on the items in the optional purchase set which are not in his shopping basket.

We believe that from the seller's perspective it is important to require a conscious attempt on part of a buyer to redeem a coupon. The objective of a couponing campaign is to provide incentive to buyers who would have otherwise not purchased the product featured in the coupon. If the coupons are redeemed automatically, they would also be redeemed for people who would have bought the product without the coupon. This essentially is lost profit for the seller.

# 7. Discussion

In the preceding sections we made several assumptions about handling coupons on the Internet. For example, we said that trading and duplication of coupons would hinder forecast of redemption rates, and that duplication and trading would have to be prevented. In this section we preset our rationale behind these assumptions.

### Do coupon duplication and trading impact forecasting of redemption rates ?

The number of paper coupons published in a magazine that will be redeemed can be calculated because the circulation of the magazine in which they are published is known and relatively stable, and past redemption rate can be used to predict the future redemption. However, with duplication, a person or family that redeemed a paper coupon once, will now redeem an electronic coupon on unpredictable number of occasions, especially trying to stock up for future.

Trading of coupons transfers the coupons to people whose characteristics, like demographics, will be completely different from the ones to whom the coupons were initially targeted. Hence the ability to predict redemption rates, even in the absence of coupon duplication, is severely diminished. In subscription based magazines, hyper linking of the content within a site as opposed to linear organization of paper magazines, will complicate the ability to use the number of subscriber log-ins to estimate the number of times a coupon is viewed.

### Is it necessary to prevent coupon duplication and trading ?

In some cases duplication and trading of a coupon are not an issue. If coupons are being used to publicize price reduction for the purpose of liquidating excess inventory, duplication and trading would be even desirable. However, if the coupons are being targeted to nonusers of a brand, trading will divert it to the users of the brand. For example, if Joe always buys brand A of coffee and Joe gets a one dollar coupon for brand B, Joe may be quite willing to sell it to a brand B user for seventy five cents, defeating the goal of brand B's manufacture.

### Should coupons be easy or difficult to use ?

Good programming instinct tells us that coupons should be extremely easy to find and use. One would even be inclined to devise programs which can automatically find coupons of products defined as desirable by a consumer, and automatically redeem them. However, stores and manufacturers will have a different point of view, especially for unlimited distribution untargeted coupons.

Using paper coupons has some degree of difficulty. The shopper has to remember to clip them, take them to the store, and then redeem them. Therefore, only a few people use them. (The redemption rate of grocery coupons is around 2 to 3 percent.) Thus, the degree of difficulty in using the coupon divides shoppers into two groups, one willing to endure the difficulty to get a lower price, and the other willing to pay the higher price to avoid the inconvenience of redeeming the coupon.

Figure 5 shows an hypothetical aggregate demand cure for a product. If no coupons are used, quantity Q2 is sold at full price P2, shown as the hashed area in Figure 5. If coupons are used automatically and thereby everybody uses them, quantity Q1 is sold as discounted price P1 resulting in revenue P1 times Q1. However, if using coupons is sufficiently difficult to prevent the buyers willing to pay the price P2 from using the coupon, but not to discourage the buyers who can afford only P1, added revenue indicated by the dotted area can be obtained. Paper coupons create this selective usage of coupons by making them somewhat cumbersome to use. Similarly, E-coupons should also require involvement on the part of the buyers, so that those willing to pay the higher price continue to do so while new buyers are attracted by the coupon.

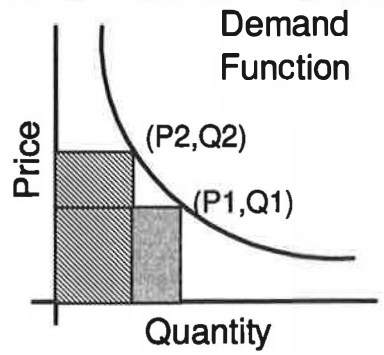


Figure 5

The above discussion also explains the fact that coupons valued at multiple dollars are in form of mail-in rebates. Redeeming them is more difficult than just clipping it off a magazine. Applying for mail-in rebates becomes very easy on the Internet, and therefore we expect that either they will be used sparingly on the Internet, or they will be turned into a more complicated process.

## 8. Summary and future work

In this paper we discussed the basic mechanisms needed to issue and redeem coupons on the Internet. We alluded to but did not cover the details of click-stream analysis and user profile, and how the two are used to effectively target potential buyers. These are integral components of a couponing application and we will be looking at them in future.

The fraud control mechanisms discussed in this paper relied on an online server to check against duplication of coupons. Off-line techniques to detect duplication is important because online techniques are invariably more expensive, and are probably not economically feasible for coupons of low monetary value. We discussed the implementation of a store coupon system. We will be extending it to support manufacturer's coupons which would require the design of client side coupon wallets, and implementation of fraud control techniques alluded to in this paper.

According to trade magazines, the coupon issuers are issuing fewer coupons compared to early nineties, and consumers are redeeming a lower percentage of them. This is attributed to the good economy, which reduces the importance of the discount offered in the coupon. Also, there is a clear shift towards issuing coupons of higher value to more narrowly targeted set of consumers.

## Acknowledgment

We are indebted to Colin Parris for creating the focus on this important topic. Most of the current implementation is by Quoc-Bao Nguyen. Sunil Noronha has provided good insights into techniques for targeting coupons.

## References:

[1] Philip Kotler, "Marketing Mamagement - Analysis, Planning, Implementation, & Control," seventh edition, 1991, Prentice Hall, ISBN 0-13-552480-6, pp. 631-639.

[2] Bruce Schneier, "Applied Cryptography,' second edition, 1996, John Wiley & Sons Inc., ISBN-0-471-11709-9, pp. 139-147.

[3] M.S. Manasse, "The Millicent protocols for electronic commerce," First USENIX workshop on Electronic Commerce, July 11-12 1995, pp. 117-123.

# General-purpose Digital Ticket Framework

Ko Fujimura and Yoshiaki Nakajima
*NTT Information and Communication Systems Labs*
{fujimura, yoshiaki}@isl.ntt.co.jp

## Abstract

A digital ticket is a certificate that guarantees certain rights of the ticket owner. There are many applications for digital tickets but the ticket properties vary depending on the application. This variety makes the digital ticket processing system expensive, especially if dedicated systems must be developed for each application. This paper thus addresses issues on developing a common data schema and processing architecture for various types of digital tickets. This paper clarifies requirements for a general-purpose digital ticket and shows four features in contrast to digital cash: 1) parameterization of ticket properties on anonymity, transferability, and divisibility; 2) machine-understandability of ticket contents; 3) state-transitionality of ticket status; and 4) composability of multiple tickets. To achieve parameterization of ticket properties and machine-understandability, we propose a Resource Description Framework (RDF)-based ticket description method. Its metadata facility enables various ticket properties to be defined using multi-layered schemata. To achieve state-transitionality and composability, we propose describing a ticket using a set of signed descriptions linked with restriction-specified incomplete/complete links. Finally, this paper proposes a set of common ticket processing components.

## 1. Introduction

A number of electronic payment schemes [1] such as encrypted credit cards [17], digital cash [8], and micropayments [5] [16] have been designed and established for Internet commerce. However, in the opposite flow of the payment, i.e., goods or products to the consumer, we depend on a physical delivery system except for a few types of digital contents, e.g., images, sounds, and computer software. Any goods that can be encoded as digital information, however, can inherently be delivered electronically.

Any business that offer services, e.g., transportation, accommodations, theaters, and restaurants, also make up a large industry and their services can be sold as tickets. A ticket is a certificate, which guarantees that the ticket owner has the right to claim the services written on the ticket. A ticket can be implemented as a digital certificate (or digital ticket) and can be delivered electronically. Although they are not common now, the business scope of electronic commerce will expand rapidly if digital tickets, which enable us to trade over the Internet, become easy to use and manage.

The number of any one particular type of digital ticket that is issued, however, would be far less compared to digital cash, because there is such a wide variety of tickets. It makes the implementation cost of the infrastructure for processing digital tickets, such as ticketing systems, ticket wallet systems, and ticket examination systems, expensive, if a system must be developed for each individual application. This paper thus addresses a ticket description method and processing architecture of general-purpose digital tickets that enable the issuing, trading, and spending of various types of digital tickets using a set of common ticket processing components.

Some digital tickets, e.g., E-Stamp [12] or e-gold [13], have already been developed. Transaction Net [19] surveys several digital values including commodity-backed money or scrip. These technologies, however, were developed for individual tickets or applications. No digital ticket framework that covers a wide range of digital tickets has been proposed yet.

This paper clarifies the requirements of general-purpose digital tickets and its four features which are not required for digital cash: 1) parameterization of ticket properties on anonymity, transferability, and divisibility; 2) machine-understandability of ticket contents; 3) state-transitionality of ticket status; and 4) composability of multiple tickets.

To establish parameters for properties and machine-understandability, this paper defines layered schemata of digital tickets using Resource Description Framework (RDF) [10]. Its metadata facility enables various ticket properties to be defined and enables the use of various ticket schemata according to the ticket type such as theater tickets or accommodation tickets. To achieve state-transitionality and composability, we propose describing a ticket using a set of signed descriptions linked with restriction-specified links. The state-transition of a ticket, e.g., payment status or reservation status, is expressed by attaching to the original ticket a description of how the status was changed. Using the restriction-specified links, flexible requirements on the description to be attached can be specified.

Finally, this paper proposes a set of common ticket processing components that can issue, trade, and spend various types of digital tickets.

## 2. Digital Ticket

In this paper, a digital ticket (simply called "ticket" hereafter) is defined as follows:

> **Definition**
>
> Let *I* be a ticket issuer, *O* be a ticket owner, and *P* be a promise to the ticket owner. A ticket is defined as Signed$_I$ (*I*, *P*, *O*), where the phrase "Signed$_I$" means that the entire block is signed by the issuer's digital signature.

Examples of promise *P* are as follows:

- A flight between Boston and Tokyo can be reserved with this ticket.
- This ticket can be exchanged for 1g of gold.
- One image file in a particular server can be downloaded with this ticket.
- After June 1998, this ticket can be exchanged for my car.
- The bearer of this ticket has unlimited telephone use for one month.

In this paper, we assume the issuance, transference, and consumption of a ticket are as follows:

### Ticket Issuance

A ticket can be freely issued bearing the ticket issuer's intention, e.g., service to be provided or commodity to be delivered. It represents a promise by the ticket issuer to complete some task or render some service. If the task described on the ticket cannot be accomplished, it will detract from the ticket issuer's credibility.

### Ticket Transference

Depending on the type of ticket, as described in Section 3, a ticket can be transferred to a third person. Typically, transferring a ticket can be accomplished using a special software component described in Section 4.4. Note that payment is usually also made for the transference of the ticket, but this payment is outside the scope of this paper. We make no assumption on the payment method.

### Ticket Consumption

A ticket is consumed typically by the issuer fulfilling the service or task represented by the ticket and the ticket being returned to or voided by the issuer or service provider. More concretely, the mechanism of this process can be represented by placing a ticket in a ticket examination machine when a service is rendered. Another example is pasting a ticket icon in an input field of a form received from some service provider (downloading an image file, etc.) to receive a service. In this process, it is assumed that the consumer agreed to void the ticket. A certificate of the consumption signed by the consumer is given to the issuer or service provider.

## 3. Related Work

Digital tickets have many similarities to digital cash. Therefore, some basic technologies for implementing digital tickets have already been developed in the research area of digital cash [8]. This section thus compares and contrasts the digital ticket to digital cash as we know it today. The following ten key features [6][7] of digital cash have been proposed.

(1) Secure (unable to alter or counterfeit)
(2) Anonymous (untraceable)
(3) Portable (physical independence)
(4) Transferable
(5) Off-line capable
(6) Divisible [7]
(7) Infinite duration (persistent)
(8) Wide acceptability (trust)
(9) User-friendly (easy-to-use)
(10) Monetary freedom (non-political) [6]

As a result of our investigation on many physical tickets, we found that required levels on (2) anonymity, (4) transferability, (6) divisibility, and (7) persistency are different according to the ticket type as shown in Table I and II. It is important for a general-purpose digital ticket architecture to process any type of ticket regardless of the required property level. Thus establishing parameters for these properties is required to cover a wide range of tickets.

In addition, we found that the three following requirements are important for digital tickets, which are not required in digital cash.

(11) Machine-understandable

Before any transaction can be conducted, the terms and description of the service or task must be objectively understood by both the service provider and consumer or owner, otherwise, the value of the ticket can not be determined. Moreover, as described in Section 4.4, this is a key property to register or search for a ticket in a marketplace when the ticket is resold.

(12) State manageable

Some tickets have a payment status, i.e., paid or unpaid, and/or reservation status, e.g., waiting list, reserved, or canceled. The status may be changed dynamically. Additionally, the ticket owner's identification that is recorded on the ticket can be rewritten when the ticket is transferred. However, it is difficult to allow these change while still guaranteeing security.

(13) Composable

Combining two or more tickets is sometimes required to obtain a service or one ticket may comprise several parts. For example, a travel ticket can comprise an accommodation ticket and a plane ticket. Also, a plane ticket can comprise a flight reservation ticket and an open ticket.

As mentioned above, digital tickets have some similar properties to digital cash but these properties vary with the ticket, and there are some new requirements not required for digital cash. A summary of this is shown in Table I.

Among the technologies to achieve the above ten properties, (1) through (10) are outside the scope of this paper since a number of methods [1][8] have already been proposed and developed. Instead, this paper proposes a ticket description method that enables the definition of various ticket properties and first establishes (11) a machine understandable ticket. Second, this paper proposes a ticket model that achieves (12) state manageable and (13) composable tickets. Finally, this paper briefly explains the prototype system of common ticket processing components that we developed.

## 4. Approach

### 4.1 Ticket description

Basic properties of a ticket comprise issuer *I*, promise *P*, and owner *O* as described in Section 2.

| Properties | | Digital Cash | Digital Ticket |
|---|---|---|---|
| (1) | Secure | Yes | Yes |
| (2) | Anonymous (Untraceable) | Yes | (2-1) Untraceable<br>(2-2) Traceable |
| (3) | Portable (Physical independence) | Yes | Yes |
| (4) | Transferable | Yes | (4-1) Transferable<br>(4-2) Not transferable |
| (5) | Off-line capable | Yes | Yes |
| (6) | Divisible (Number of times to be consumed) | Yes | (6-1) Specified times<br>(6-2) Only once<br>(6-3) Infinite times |
| (7) | Persistent (Valid period) | Yes | (7-1) Specified period<br>(7-2) Persistent |
| (8) | Wide acceptability | Yes | Yes |
| (9) | User-friendly | Yes | Yes |
| (10) | Monetary freedom | Yes | No |
| (11) | Machine-understandable | No | Yes |
| (12) | State manageable | No | Yes |
| (13) | Composable | No | Yes |

**Table I. Digital Ticket Properties**

| Examples | Anonymity[1] | Transferability[1] | Number of times |
|---|---|---|---|
| Event ticket | Yes | Yes | Only once |
| Plane ticket | No | No | Only once |
| Lottery ticket | Yes | No | Only once |
| Stamp | Yes | Yes | Only once |
| Telephone card | Yes | Yes | Specified |
| Cash | Yes | Yes | Specified |
| Software license | No | Yes | Infinite |
| Transportation pass | Yes | No | Infinite |
| Gate card | No | No | Infinite |
| Driver's license | No | No | Infinite |

[1] It depends on the specific ticket. This table only shows the tendency for the ticket types.

**Table II. Properties of Specific Ticket Types**

Transferability, anonymity, number of times to be consumed, and valid period shown in Table I are also important properties that vary depending on the ticket and determine how the ticket should be processed. Regarding the anonymity property, however, it can depend on how owner *O* is specified. For example, anonymity can be achieved by using a pseudonym [3][4] to specify the owner and keeping the real name secret except to the pseudonym issuer. We therefore assume that the anonymity property is subsumed by the owner property. To display or print the ticket, a view property is also needed. All of these properties can be and must be specified regardless of the ticket type.

There are properties determined by each ticket type, e.g., event ticket, plane ticket, accommodation ticket, and software license. Examples of these properties are flight number or departure date, etc. These properties are defined by each industry.

In addition, there are properties that are defined by each issuing company or individual. An example of these properties is mileage points. Ticket properties are thus classified into the following three schema layers.

Layer 1
Common ticket properties that do not depend on the ticket type:

- Issuer
- Promise (Details are defined in upper layers)
- Owner (incl. pseudonym)
- Transferability
- Number of times to be consumed
- Valid period
- View
- Issuer's signature on above

Layer 2
Common ticket properties defined by each industry

Layer 3
Any ticket property defined by each issuing company or individual

We examined existing data describing technologies to see if they can be used to describe various ticket properties and if they can use multi-layered schemata. We found that Resource Description Framework (RDF) [10] developed by W3C fulfills the requirements above. RDF is a foundation for exchanging machine-understandable information on the Web and no application-specific assumption is made. We therefore defined ticket-specific schema layers based on RDF.

Figure 1 shows an example of describing a concert ticket in the RDF syntax. Note that the RDF specification is a draft specification and may be updated or replaced.

In RDF, the meaning and restrictions of the properties used in an RDF description must be defined in the RDF schema, which are defined somewhere in the network. The RDF schema is defined in an RDF schema specification [11]. See details in the specification.

Using XML [9] namespace facility, which may be contained in the RDF document's prolog, Universal Resource Identifiers (URIs) for layered ticket schemata can be referred to as described in this example. In this example, it is assumed that each schema is distributed and their URIs are defined as follows:

Layer 1
```
<?xml:namespace
   name="http://tickets.org/schemas/ticket#"
   as="DTK"?>
```

Layer 2
```
<?xml:namespace
   name="http://events.org/schemas/event-
   ticket#" as="EVT"?>
```

Layer 3
```
<?xml:namespace
   name="http://mycorp.com/schemas/my-ticket#"
   as="MTK"?>
```

This facility makes it easy to specify and maintain the schemata defined by different organizations. This approach, thus, not only achieves (11) the machine-understandability feature described in Section 3, but also has the advantage that many organizations can easily define their own ticket schema.

As shown in Figure 1, the `<DTK:Issuer>`, `<DTK: Promise>`, `<DTK:Owner>`, `<DTK:Transferability>`, `<DTK: NumberOfTimes>`, `<DTK:ValidPeriod>`, `<DTK:View>`, and `<DTK:Signature>` tags are used to define the layer 1 properties.

The Conditions property in Figure 1 describes detailed contract conditions. Such information is not always necessary but the size is not negligible for circulating a ticket, especially when the ticket is recorded on a smart card. In this case, the description can be located on a certain server on the network and referred to by the link if necessary. It also reduces communication cost. The View property, which defines the image data of the ticket, is also defined using a link as shown in Figure 1.

In RDF, the Description element itself creates a resource and it can be referred to by the bagID. Using the bagID, the scope of the definition that the signature is applied to can be specified. See details in the specification [10].

## 4.2 Restriction-specified incomplete link

In this section, we propose an approach to implement the (12) state-manageable ticket features. State-manageability enables the status of the ticket, e.g., owner identifier, payment status, or reservation status, to change dynamically.

However, it is not easy to implement because the ticket is signed by the issuer, and nobody except the issuer can

```
<?xml:namespace name="http://mycorp.com/schemas/my-ticket#" as="MTK"?>
<?xml:namespace name="http://events.org/schemas/event-ticket#" as="EVT"?>
<?xml:namespace name="http://tickets.org/schemas/ticket#" as="DTK"?>
<?xml:namespace name="http://www.w3.org/TR/WD-rdf-syntax#" as="RDF"?>

<RDF:RDF>
  <RDF:Description ID="ticket_001" bagID="ticket_bag_001">
     <DTK:Issuer>issuer@mycorp.com</DTK:Issuer>
     <DTK:Promise>
        <RDF:Description>
           <EVT:Reference>R02-345</EVT:Reference>
           <EVT:Name>FooBar Concert</EVT:Name>
           <EVT:Type>Adult</EVT:Type>
           <EVT:Place>New York City Hall</EVT:Place>
           <MTK:Points>20</MTK:Points>
           <MTK:Conditions href="http://mycorp.com/conditions"/>
        </RDF:Description>
     </DTK:Promise>
     <DTK:Owner>u1@host1.com</DTK:Owner>
     <DTK:Transferability>ANYBODY</DTK:Transferability>
     <DTK:NumberOfTimes>ONCE</DTK:NumberOfTimes>
     <DTK:ValidPeriod>1998-2-20</DTK:ValidPeriod>
     <DTK:View href="http://mycorp.com/my-ticket.gif"/>
  </RDF:Description>

  <RDF:Description href="#ticket_bag_001">
     <DTK:Signature>05b8cfc04d05a8cfc03d2549cfc03d36</DTK:Signature>
  </RDF:Description>
</RDF:RDF>
```

**Figure 1. A concert ticket example in RDF**

alter the contents of the ticket. However, it is often required to change the contents of the ticket with the approval of the original ticket issuer or a Trusted Third Party (TTP), etc.

To express the state-transition of ticket properties without changing the original ticket definition, we propose defining a ticket using a linked set of signed descriptions, i.e., the state-transition of a ticket property is defined by attaching a state-transition description linked from the property definition in the original description:

### Original Description:

```
<RDF:RDF>
  <RDF:Description ID="ticket_001"
                   bagID="ticket_bag_001">
    ...
    <Prop1/>
      <RDF:Description>
         <DTK:Value>current-value</DTK:Value>
         <DTK:NewValue href="#prop1_001"/>
      </RDF:Description>
    </Prop1>
    ...
  </RDF:Description>
  <RDF:Description href="#ticket_bag_001">
    <DTK:Signature>...</DTK:Signature>
  </RDF:Description>
</RDF:RDF>
```

Assume that the value of property Prop1 is changed from "current-value" to "new-value," then the following description is attached to the above:

### Attached Description:

```
<RDF:RDF>
  <RDF:Description ID="prop1_001"
                   bagID="prop1_bag_002">
     <DTK:Value>new-value</DTK:Value>
  </RDF:Description>
  <RDF:Description href="#prop1_bag_002">
     <DTK:Signature>...</DTK:Signature>
  </RDF:Description>
</RDF:RDF>
```

In the above examples, link ID "prop1_001" must be defined and signed before the referred description is defined. We call such a link an *incomplete link.*

The link ID must be universally unique, since the attached signed description could be used for malicious purposes. An URI can be used for that purpose. Note that short IDs are used in this paper for readability.

The incomplete link shown in the above example, allows any description to be located as the destination of the link without restriction. It is inconvenient especially when the referred description must be signed (or issued) by the original issuer or others who have been delegated the right to define the value. To fulfill this requirement, this paper proposes a new concept called *restriction-specified incomplete link* that restricts only the valid description so that it can be located on the destination of the incomplete link. If an invalid description is located, it will be detected by the validation check system.

An example of a restriction-specified incomplete link is as follows:

**Original Description:**

```
<RDF:RDF>
  <RDF:Description ID="ticket_001"
                   bagID="ticket_bag_001">
    ...
    <Propl>
      <RDF:Description>
        <DTK:Value>current-value</DTK:Value>
        <DTK:NewValue href="#propl_001"/>
        <DTK:Restriction>
         <RDF:Description>
          <DTK:Issuer>u2@host.com</DTK:Issuer>
         </RDF:Description>
        </DTK:Restriction>
      </RDF:Description>
    </Propl>
    ...
  </RDF:Description>
  <RDF:Description href="#ticket_bag_001">
     <DTK:Signature>...</DTK:Signature>
  </RDF:Description>
</RDF:RDF>
```

This example restricts the description of propl_001 to be attached to the description of issuer u2@host.com. An example of the valid description is as follows:

**Attached Description:**

```
<RDF:RDF>
  <RDF:Description ID="propl_001"
                   bagID="propl_bag_002">
    <DTK:Issuer>u2@host.com</DTK:Issuer>
    <DTK:Value>new-value</DTK:Value>
  </RDF:Description>
  <RDF:Description href="#propl_bag_002">
    <DTK:Signature>...</DTK:Signature>
  </RDF:Description>
</RDF:RDF>
```

As we described in the above examples, a restriction-specified incomplete/complete link is defined as follows:

> **Definition**
>
> Let $D_0$ be a signed description, $P$ be a property in $D_0$, $V$ be the current value for $P$, $D_1$ be a signed description to be attached, $L$ be a link to $D_1$, and $R$ be a restriction for $D_1$. A *restriction-specified incomplete/complete link* is defined as a tupple of ($P$, $V$, $L$, *and* $R$) .
>
> The value of $P$ is interpreted as $D_1$ if $D_1$ is instantiated and $D_1$ satisfies restriction $R$, otherwise the value of $P$ is interpreted as $V$., where we call $D_1$ *instantiated* if $D_1$ is located on destination $L$.

As shown in the above examples, the `<Propl>`, `<DTK:Value>`, `<DTK:NewValue>`, and `<DTK:Restriction>` tags are used to define $P$, $V$, $L$, and $R$ respectively.

Restriction $R$ can be any restriction, but the following three types of restrictions would be enough to describe most tickets based on our investigation.

**Schema restriction:** The schema (or format) of the attached description. It defines properties to be defined and its necessity property, i.e., mandatory or optional.

**Property value restriction:** The value for the specific property of the attached description. Note that property value restriction can also be defined within the schema of the property if the schema is not assumed to be shared.

**Hash value restriction:** The hash value of the description. Note that this restriction can only be applied when the attached description has been instantiated when this restriction is specified.

## 4.3 Composable ticket model

In this section, we propose an approach to implement the (13) composable ticket features. Composability enables the definition of a complex ticket using multiple sub-tickets.

There are many cases when a sub-ticket must be issued separately with the original ticket typically because the tickets are issued by different organizations or issued at different times.

The restriction-specified incomplete/complete link can also be applied to composable tickets. The signed description to be attached can be a ticket.

For example, suppose that a plane ticket can comprise an open ticket and a flight reservation ticket, which are used for a certain air route and reserved for a certain flight on the reserved date, respectively. The reservation ticket can be considered as a sub-ticket of the open ticket and attached to the reservation status property of the open plane ticket.

```
<RDF:RDF>
  <RDF:Description ID="ticket_001" bagID="ticket_bag_001">
    <DTK:Issuer>issuer@airline1.com</DTK:Issuer>
    <DTK:Promise>
      <RDF:Description>
        <MTK:Departure>London</MTK:Departure>
        <MTK:Destination>Boston</MTK:Destination>
        <MTK:Reservation>
          <RDF:Description>
            <DTK:Ticket href="#ticket_002"/>
            <DTK:Restriction>
              <RDF:Description>
                <RDF:InstanceOf href="http://airline1.com/schema#RsvTicket"/>
                <DTK:Issuer>issuer@airline1.com</DTK:Issuer>
              </RDF:Description>
            </DTK:Restriction>
          </RDF:Description>
        </MTK:Reservation>
        <MTK:Conditions>
          <RDF:Description>
            <DTK:Value href="http://airline1.com/conditions"/>
            <DTK:Restriction>
              <RDF:Description>
                <DTK:Digest>476169572bb3680708cd4204907735ac</DTK:Digest>
              </RDF:Description>
            </DTK:Restriction>
          </RDF:Description>
        </MTK:Conditions>
      </RDF:Description>
    </DTK:Promise>
    <DTK:Owner>u1@host1.com</DTK:Owner>
    <DTK:Transferability>NO</DTK:Transferability>
    <DTK:NumberOfTimes>ONCE</DTK:NumberOfTimes>
    <DTK:ValidPeriod>INFINITE</DTK:ValidPeriod>
    <DTK:View href="http://airline1.com/plane_ticket.gif"/>
  </RDF:Description>

  <RDF:Description href="#ticket_bag_001">
    <DTK:Signature>9572bb3680708476735ac16cd4204907</DTK:Signature>
  </RDF:Description>
</RDF:RDF>
```

**Figure 2. Open plane ticket example**

Figure 2 shows an example of describing an open plane ticket in the RDF syntax. In this example, the two restrictions of a sub-ticket to be attached are defined.

- The ticket must be an instance of the reservation ticket. This is an example of schema restriction.
- The ticket must be issued by issuer@airline1.com. This is an example of a property value restriction.

The Conditions property in Figure 2 also includes the hash value restriction described in Section 4.2.

The `<RDF:InstanceOf>` and `<DTK:Digest>` tags are used to define schema restrictions and hash value restrictions respectively. To define a property value restriction, the tag for the property, i.e., the `<DTK:Issuer>` tag in this example, is used in the description of `<DTK:Restriction>`.

There are many applications of composable tickets as shown in Table III. It is possible to change the owner property of a transferable ticket by attaching a transfer (ticket) if a restriction-specified link is defined in the owner property. In this case, the restriction is that issuer of the transfer is the transferor.

It is also possible to control anonymity of a ticket by attaching different types of public key (PK) certificates to the owner property of the ticket. That is, if a ticket must be traceable, a PK certificate with user identifier is attached, and if a ticket must be untraceable, a PK certificate without user identifier is attached.

Other examples are a composition of a deferred payment ticket and a check, or an authorized document and the approval stamp as shown in Table III.

## 4.4 Common processing components

This section explains the prototype system of common ticket processing components that we developed.

| Original ticket | | Attached ticket / description | |
|---|---|---|---|
| Type | Property | Schema restriction | Value restriction |
| Any transferable ticket | Owner | Transfer (certificate) | Issuer is the transferor |
| Any traceable ticket or transfer attached | Owner | PK certificate with user identifier | Issuer is an CA |
| Any untraceable ticket or transfer attached | Owner | PK certificate without user identifier | Issuer is an CA |
| Any deferred payment ticket | Payment status | Check or draft | Issuer is a bank |
| Any documents to be authorized | Approval | Approved stamp | Issuer is the specified issuer |
| Any ticket details/conditions are described | Conditions | None | Digest value is specified (See Figure 2) |

**Table III. Application of Composable Tickets**

### (1) Ticket Editor

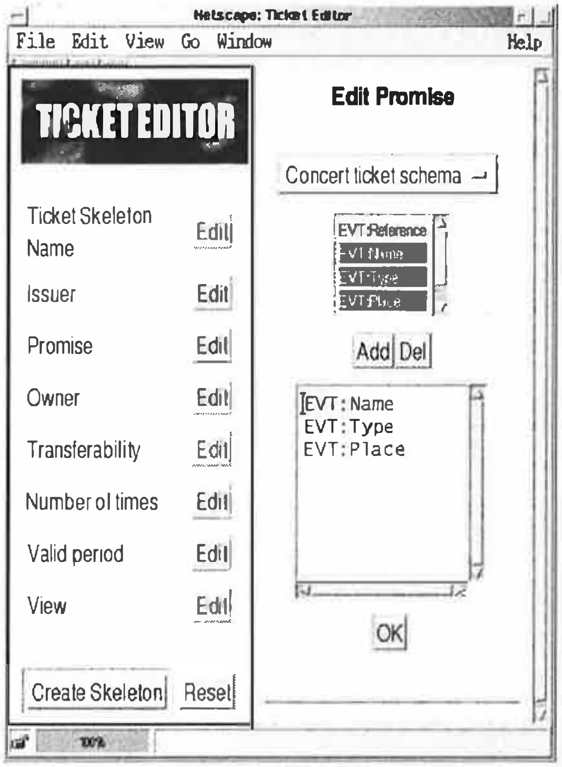
The *ticket editor* is a component that generates a ticket skeleton by interacting with the ticket issuer and using pre-defined ticket schemata (layer 1, 2, and 3). A *ticket skeleton* is similar to a ticket except that it may leave some properties blank, e.g., reference number, required tickets, or the signature of the issuer. This is because these property values are usually given dynamically. The GUI of the ticket editor is shown in Figure 3. This component is typically used by ticket designers. Note that defining the ticket schemata is out of scope of this paper.

### (2) Ticket Generator

The *ticket generator* is a component that generates ticket instances from a ticket skeleton by interacting with users or customers to whom the ticket is issued. The ticket generator fills in the blank properties by generating or prompting input from the users and signs the ticket. This component is typically used in a Web server application. Figure 4 shows the flow of how the ticket is generated using the ticket editor and generator.

### (3) Ticket Reader

The *ticket reader* is a component that checks the validity of the ticket and voids the ticket when a service is rendered. The voiding method depends on the type of the ticket. For example, it can be done by auditing a record in which the ticket owner acknowledges the ticket consumption and signs it This tool is typically used in ticket examination machines



EVT:Name
EVT:Type
EVT:Place

**Figure 3. Ticket editor**

### (4) Ticket Transferor

The *ticket transferor* is a component that changes the ticket owner identity to a new person. Ticket transfers can be done by adding a transfer ticket that specifies the new ticket owner as described in Sections 4.2 and 4.3. This tool is typically used in the transferor's (or ticket owner's) ticket wallet or server.

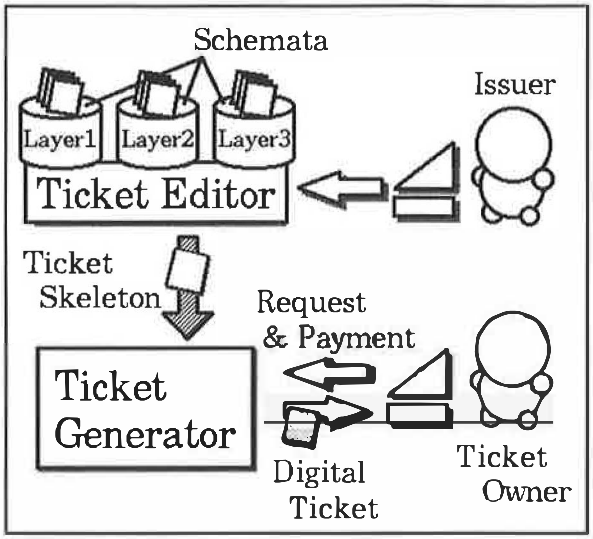


**Figure 4. Generating a ticket**

### (5) Ticket Viewer

The *ticket viewer* is a component that displays the contents (or meaning) of the ticket and checks the validity of the ticket. This component is typically used in the purchaser's ticket wallet and used to confirm the contents of the ticket before conducting a transaction. This reduces the number of problems encountered while trading because ticket purchasers can confirm the contents of the ticket before making a transaction.

Details regarding the processing scheme of these components are out of the scope of this paper. The implementation details will be presented in another paper.

Our current implementation uses an online checking scheme to prevent duplicate redemption. However, more sophisticated protocols [2][7] and/or smart card protocols [14][15] invented for electronic cash can be used if more complete anonymity, transferability, divisibility, and off-line capabilities are required. This paper does not conflict with these protocols, instead one of our goals is independence from the infrastructure for preventing duplicate redemption. It is important to have numerous types of tickets but common components or systems for ticket processing can be shared by establishing parameters for the layer 1 properties as described in this paper.

Additionally, a seller (or reseller) of a ticket can transmit the ticket information to a marketplace without typing it in. The process can be completed by click- or drag & drop-based GUIs. Also, the contents described in the ticket can be read mechanically, thus making the purchaser's search for tickets in the marketplace very efficient. This can be expected to promote resale or recycling over the Internet. Figure 5 shows the outline of a ticket marketplace.

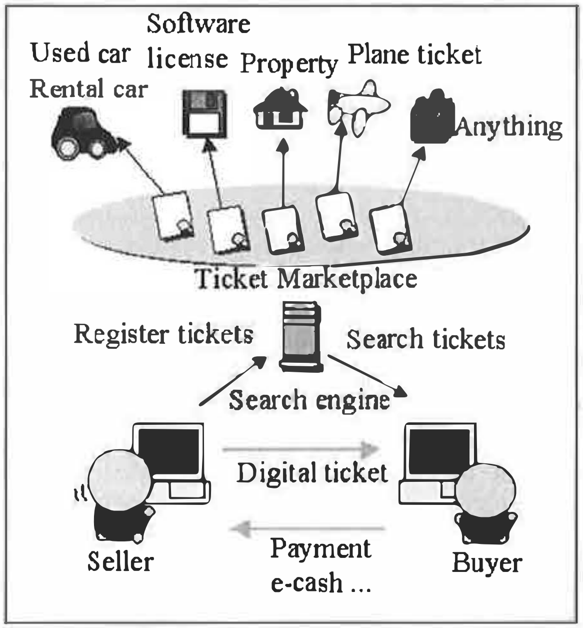


**Figure 5. Ticket marketplace**

## 5. Conclusions

This paper classified various types of digital tickets and clarified common properties of digital tickets.

A ticket description method was proposed that enables various ticket properties to be defined. The tickets described by the proposed method are machine understandable, state-manageable, and composable. To achieve machine understandability, a ticket using RDF with three-layer ticket schemata was described in Section 4.1. To achieve state-manageability, a new ticket description method that uses restriction-specified incomplete link was proposed in Section 4.2. To achieve composability, a composable ticket model that expresses a composite ticket using a set of required subtickets was proposed in Section 4.3.

Common components that can be used in the issuing, trading, and spending of various types of digital tickets were presented in Section 4.4. These components can be shared among applications because ticket properties of the first layer schema determine the processing patterns regardless of application-specific ticket properties of the second layer schemata defined by each industry.

A common processing platform for digital tickets makes it easy for a small enterprise or individuals to issue or trade their own tickets over the Internet, even though the number of tickets that are issued is small.

There is possibility that all kinds of tickets, including train tickets and admission tickets to amusement parks, will be substituted with digital tickets, and that the ticketing machines will be substituted with Web terminals when smart cards [18] become more popular. To make it practical, a standard ticket description method and ticket processing infrastructure must be established and we believe that our framework presented here is the key.

## Acknowledgements

We would like to thank Yuji Takehisa, Jun Sekine, Yasunao Mizuno, Kazuo Matsuyama, Masayuki Terada, Hiroshi Kuno, Ayumu Eto, and the referees for their suggestions.

## References

[1] N. Asokan, P. A. Janson, Michael Steiner, and M. Waidner, "The State of the Art in Electronic Payment Systems," *IEEE Computer*, Sep. 1997, pp. 28-35.

[2] D. Chaum, "Privacy Protected Payments Unconditional Payer and/or Payee Untraceability," In *Smart Card 2000*, North-Holland, Amsterdam, 1989, pp. 69-93.

[3] G. Davida, Y. Frankel, Y. Tsiounis, and M. Yung, "Anonymity Control in E-Cash Systems," In *Proceedings of Financial Cryptography '97*, LNCS 1318, pp. 1-16.

[4] Eran Gabber, Phillip B. Gibbons, Yossi Matias, and Alain Mayer, "How to Make Personalized Web Browsing Simple, Secure, and Anonymous," In *Proceedings of Financial Cryptography '97*, LNCS 1318, pp. 17-31.

[5] S. Glassman, M. Manasse, M. Abadi, P. Gauthier, and P. Sobalvarro, "The Millicent Protocol for Inexpensive Electronic Commerce," In *Proceedings of WWW4*, http://www.w3.org/Conferences/WWW4/Papers/246/

[6] J. W. Matonis, "Monetary Freedom," In *Proceedings of INET95*, Internet Society, http://info.isoc.org/HMP/PAPER/136/html/paper.html

[7] T. Okamoto and K. Ohta, "Universal Electronic Cash," In *Advances in Cryptology, Proceedings of CRYPTO '91*, J. Feigenbaum (Ed.), LNCS 576, pp. 324-337.

[8] Peter Wayner, "Digital Cash," Academic Press Ltd., 1997.

[9] Extensible Markup Language (XML) specifications, The World Wide Web Consortium, 1998, http://www.w3.org/XML/

[10] Resource Description Framework (RDF) Model and Syntax, The World Wide Web Consortium, Working Draft, 1998, http://www.w3.org/TR/1998/WD-rdf-syntax-19980216

[11] Resource Description Framework (RDF) Schemas, The World Wide Web Consortium, Working Draft, 1998, http://www.w3.org/TR/WD-rdf-schema

[12] E-Stamp Corporation, "E-Stamp," http://www.e-stamp.com/

[13] Gold & Silver Reserve, Inc., "e-gold," http://www.e-gold.com/

[14] Java Card Forum, http://www.javacardforum.org/

[15] MONDEX International Ltd., http://www.mondex.com/

[16] The NetBill Electronic Commerce Project, http://www.ini.cmu.edu/netbill

[17] Secure Electronic Transaction (SET) specifications, http://www.mastercard.com/set/

[18] Smart Card Forum, http://www.smartcrd.com/

[19] Transaction Net, "Complementary Currencies and Exchange Networks," http://www.transaction.net/money/comp/

# Towards A Framework for Handling Disputes in Payment Systems

N. Asokan, Els Van Herreweghen, Michael Steiner
*IBM Zurich Research Laboratory*
*8803 Rüschlikon, Switzerland*
{aso,evh,sti}@zurich.ibm.com

## Abstract

The ability to support disputes is an important, but often neglected aspect of payment systems. In this paper, we present a language for expressing dispute claims in a unified manner, independent of any specific payment system. We illustrate how to support claims made in this language with evidence tokens from an example payment system. We also describe an architecture for dispute handling. Our approach may be generalised to other services where accountability is a requirement.

## 1 Introduction

### 1.1 Importance of Dispute Handling in Electronic Commerce

Services in an electronic commerce system involve more than one player. An effective system guarantees that if all players behave correctly ("honestly") according to some pre-defined protocol, each player obtains the services it expects. A system with *integrity* [12] guarantees that, in addition, honest players are also protected against the incorrect behaviour of other players they do not trust. For example, in a payment system, an integrity requirement of an honest payer is that the payee receives at most the amount of value authorised by the payer.

Sometimes practical considerations may render it desirable to settle for a weaker form of integrity — the integrity requirement is modified as follows: even if the system is not able to prevent a dishonest player from causing "harmful" effects to the honest player(s), it must allow the honest player(s) to later prove the behaviour of the dishonest player. "Standing order" payments is an example. The payer instructs his bank to allow periodic value transfers requested by the payee (e.g., for paying utility bills). While the payer cannot prevent the payee from requesting more money than necessary (e.g., more than the amount of the monthly bill), he can prove the amount transferred by obtaining a statement from the bank.

Furthermore, electronic commerce transactions typically have legal significance in the real world. This means that even if a transaction is concluded successfully, there may be subsequent disputes about what happened during the transaction (or whether in fact an alleged transaction took place).

Thus, the ability of an honest player to win any dispute about a past or current transaction is often an important requirement.

### 1.2 Handling Disputes

Even those systems which have accountability as one of their major goals (signature systems such as RSA [13], payment systems such as SET [10], or integrated electronic purchase systems such as NetBill [6]), usually limit themselves to the generation and collection of evidence. Analysis of these systems may include proofs which demonstrate that the collected evidence is enough to win any disputes. It is assumed that such evidence can be used in some dispute resolution procedure external to the system. Obviously, this is not practical: it excludes the possibility of the system making its own decisions based on the outcome of disputes, or internally trying to recover from errors or failures (e.g., caused by loss of a network connection).

This work was partially supported by the Swiss Federal Department for Education and Science in the context of the ACTS Project AC026, SEMPER; however, it represents the view of the authors. SEMPER is part of the Advanced Communications Technologies and Services (ACTS) research program established by the European Commission Directorate General XIII. For more information on SEMPER, see `http://www.semper.org`

Moreover, evidence tokens are essentially part of the internal structure of the system; their structure and raw contents are not relevant outside the system. For instance, a payment receipt in the form of a digital signature is outwardly just a string of bits. Even if the receipt is in a format which allows anyone to securely verify who signed it, and when it was sent or received, the semantics to the evidence have to be added by the system itself. Outside the system (that is, from the point of view of the user of a system), what is necessary is to know what the evidence *means*, and how it can be *used* in disputes. Thus, a system providing a primary service (e.g., a payment service) should also support a *dispute service*. The dispute service specifies how to initiate and resolve disputes for the given primary service.

In a dispute, there is a set of (one or more) players called *initiators* who start the dispute and another set of players called *responders* who participate in it. A special player called the *verifier* or *arbiter* makes, or helps in the making of, the final decision regarding disputes, according to some well-defined procedures which can be verified by anyone. The initiator(s) try to convince the verifier of a *claim*. Initiators may support their claims by producing evidence or engaging in some sort of a proof protocol. Responders may attempt to disprove the claims. The verifier analyses the claims made and the evidence presented. This analysis may lead to a judgment as to whether a dispute claim is valid or not.

Completely automated dispute resolution may not always be feasible or even desirable. Our dispute handling service should instead be used as a tool in human-driven dispute resolution. For example, it can be used by an expert witness in court in order to support his testimony. Or, it can be used by an entity like the Online Ombuds Service [8] which is not a legally competent authority but helps players resolve their disputes. In these cases, the verifier does not make a final decision. Instead, it presents an analysis of the evidence to a human judge. The verifier may even be one of the players themselves, trying to settle a dispute in a friendly way without going to court, or trying to convince itself of which disputes it can win.

The first step in developing a coherent approach to dispute handling is to figure out how to define a dispute handling service given the description of some primary service. To keep the problem tractable, we focus on handling disputes in payment systems - we will however attempt to stay general as far as possible so that the approach outlined here has the potential of being applicable to other types of generic service definitions as well. Our goal is to stay independent of any specific payment system. This approach is consistent with existing efforts to define generic payment services [1, 14] which enable users (human users or applications acting on their behalf) to invoke payment services in a system-independent fashion. In trying to preserve the same generality in the definition of the dispute service, we aim at developing a unified framework integrating both payment and dispute services.

In Section 2, the problem of how to express dispute claims is studied. In Section 3, the problem of how to map evidence tokens to dispute claims is investigated. This mapping is payment system-specific and needs to be done internally in every payment system. As an example, we will use a simplified version of the iKP [3] payment protocol.[2] An overall architecture for dispute handling, including a general dispute resolution protocol, is also described.

## 2 Expressing Dispute Claims

### 2.1 What to Dispute?

Consider a payment system which implements the services defined by a generic payment service (e.g., the one described in [1]). The primary purpose of the generic payment service is the transfer of value from payer to payee.

To make a value transfer, the payer tells the system who the payee is, what amount is to be transferred, and certain other parameters.

- pay $200 to BobAir ("#434: for flight 822 on Jan 19").

In a generic payment service, and using the ISO-OSI approach of modelling a distributed system [9], this may be represented by a service primitive pay which could take the following pieces of information as parameters: payee, amount and ref (an external reference string enabling the payment transaction to be linked to an external context). In order to complete the value transfer, the payee invokes a receive primitive with input parameter ref (optionally also payer and amount) and output parameters payer, amount. Figure 1 illustrates the interface events during a payment.

[2] SET is the proposed standard for credit-card payments on the Internet. However, we will use iKP here since its simplicity helps illustrate the approach more clearly.

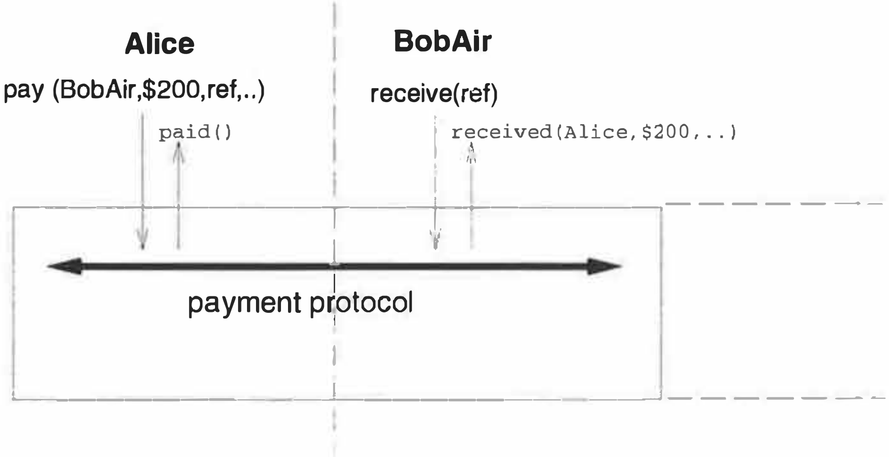


Figure 1: An Example Payment Transaction

Before formally defining a language to express dispute claims, let us attempt to get an idea of the kinds of claims that need to be expressed. What sorts of disputes, related to the above value transfer, does the payer (Alice) expect to be able to initiate and win? For example, Alice may want to claim that

- she paid $200 to BobAir (perhaps because BobAir refused to send the tickets claiming no payment was made), or
- her payment was made before Jan 12 (perhaps because there was a deadline).

Some disputes may be about negative claims: for example, BobAir may want to prove that

- BobAir did **not** receive $200 from Alice.

In other words, dispute claims are statements about the characteristics of value transfer. These characteristics are determined by the service primitives used, together with their parameters, and additional contextual information (such as the time of value transfer).

In addition to the payer and payee, a financial institution may be involved in creating a digital representation of money or converting it back to real value. Thus the value transfer may involve two or more sub-protocols involving different pairs of players. For example, in a *cheque-like* [1] model, the payer sends a "form" (e.g., a cheque or a credit card slip) to the payee using a payment protocol and the payee may use a deposit or capture protocol to claim the real money. This leads to two other types of dispute claims.

- Suppose BobAir makes an offer to Alice for a cheap ticket if she made the payment before Jan 12. Alice goes through the steps of the payment protocol (e.g., sending a credit-card slip). However, BobAir changes his mind after receiving the credit-card slip — he does not "capture" Alice's payment. Alice cannot of course prove that the value transfer took place. But if she has a signed acknowledgement from BobAir, she can prove that the value transfer *could* have taken place without further help from Alice, if BobAir had wanted.
- Suppose Alice pays $200 to BobAir using a debit card. Later, she finds an entry in her monthly statement indicating a debit of $300. Alice may now want to start a dispute with the bank claiming that she approved a debit of only $200. In other words, a single original transaction could lead to two different types of disputes: one involving the payer and the payee, and the other involving the payer and the bank.

## 2.2 Value Transfers as Primitive Transactions

Users expect a system to provide a certain service. Therefore, disputes in the system are about an instance of the service that was or could be provided. The primary service provided by the generic payment service is value transfer from one player to another. The model in [1] assumes four types of players involved in value transfer: the payer, the payee, the issuer, and the acquirer. Value transfers between the issuer and acquirer are carried out over traditional banking systems; this transfer is outside the scope of the generic payment service. Thus, for the purpose of our disputes, we will consider the issuer and acquirer as a single entity, called the *bank*.

As shown above, we also need to express and conduct disputes about transfers between payer or payee and bank. This requirement leads us to follow the approach taken in [11] of defining three different types of value transfer as shown in Figure 2.

- In *value subtraction*, a user allows the bank to remove "real value" from the user; this implies the user's right to spend "electronic value."
- In *value claim*, a user requests that the bank gives "real value" to the user.
- In *payment*, the payer transfers value to the payee.

We now define a *primitive transaction* as an instance of one of these value transfers. A primitive transaction represents a partial *view* of a subset of the players on the overall transaction.

In some cases, a primitive transaction actually corresponds to a protocol run of the underlying payment system. For example, a withdrawal protocol run in a cash-like system is a *value subtraction* primitive transaction. Thus, *value subtraction* completes independently of and before the actual payment. The *payment* and *value claim* complete after the actual cash payment protocol is run and the payee has deposited the coins with the bank.

In other cases, the primitive transaction is a purely *virtual* one and represents only the *view* of a subset of the players. For instance, a payment protocol run in a cheque-like payment system is seen by the set {payer, bank} as a *value subtraction* primitive transaction while it is seen by the set {payer, payee} as a *payment* primitive transaction.

Given the definition of a generic payment service, one can make a mapping between its service primitives on the one hand, and the primitive transactions (payment, value subtraction, value claim) effectuated by those primitives on the other hand. We refer to [2] for a description of such a mapping for the generic payment service described in [1].

Rather than referring to specific protocols or service primitives, we will state dispute claims in terms of whether or not the service defined by a primitive transaction did (or could) take place. If a value transfer is reversed (e.g., the payee refunded the payer), it is equivalent to the value transfer not having taken place at all. However, it must still be possible to express a claim like "Alice did pay $200 to BobAir in the past" which must be true, even if the payment was later refunded by BobAir.

## 2.3 Statements of Dispute Claims

### 2.3.1 Syntax

We express a statement of dispute claim as a formula in a first-order logic with certain modal extensions. The language of the logic consists of the following symbols: logical connectives, typed variables, typed constants, and relational connectives (which are functions of variables/constants of the appropriate type).

There are three types of variables: primitive transaction (pt), roles, and attributes. The pt variable can take its value from a well-defined enumerated set. In the case of the payment service, this set consists of PAYMENT, VALUE_SUBTRACTION, and VALUE_CLAIM. Each primitive transaction has a set of well-defined *role variables* associated with it. For example, PAYMENT has payer and payee as associated role variables. The role variables belong to a type called *id_val*, which represents distinguished names according to some well-defined naming scheme (e.g., account numbers or certified e-mail addresses). Each primitive transaction also has a well-defined set of *attribute variables* associated with it. For simplicity, we assume that all value transfer primitive transactions have the same set of attribute variables: amount, time,[3] and ref. The attribute variables are typed — they take their values from the appropriate domains. In the case of the payment service, we assume that amount, time, and ref take values from domains named *amount_val, time_val, and ref_val* respectively. Each attribute, depending on its type, has a finite set of relational operators associated with it. Table 1 lists the variables in the generic payment service, their domains, and applicable relational operators. We also allow two logical connectives: $\wedge$ (conjunction) and $\neg$(negation), a parenthetical operator for specifying precedence, and modal operators called **can_without**, **could_without**, **once**, **always** and

[3] There may be several different timestamps involved; for simplicity, we assume that there is only one instant at which the transaction is considered to have taken place.

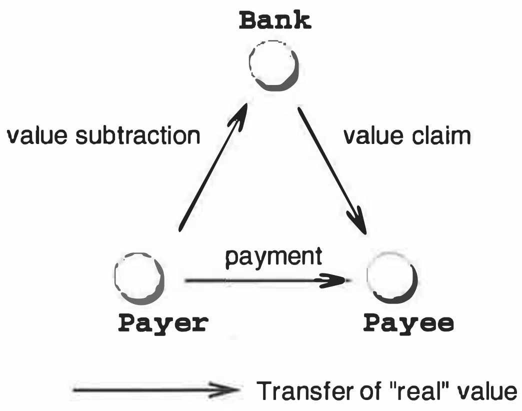


Figure 2: Value Transfer Transactions

| variable | domain | relational operators |
|---|---|---|
| pt | {PAYMENT, VALUE_SUBTRACTION, VALUE_CLAIM} | $=$ |
| <*role*> | *id_val* | $=$ |
| amount | *amount_val* | $<\ \leq\ =\ \geq\ >$ |
| time | *time_val* | $<\ \leq\ =\ \geq\ >$ |
| ref | *ref_val* | $=$ |

Table 1: Attributes and Operators of Primitive Transactions

**never**. A comma indicates concatenation.

The rules to construct valid dispute claim statements are described in the grammar specifications shown in Table 2. Note that this grammar is only payment specific in its concretisation of possible values for pts, roles, attributes and relops, and as such represents one instantiation of a family of grammars, each instantiation of which defines a grammar for dispute statements related to a specific service (payment, non-repudiation, etc.). From now on, we will simply write PRIMITIVE_TRANSACTION_NAME to denote the predicate 'pt=PRIMITIVE_TRANSACTION_NAME.' Also, when a conjunction ($\wedge$) is obvious, we omit it. For example, 'PAYMENT payer =Alice payee =BobAir' is shorthand for 'pt=PAYMENT $\wedge$ payer =Alice $\wedge$ payee =BobAir'.

### 2.3.2 Semantics

First, let us try to capture the intuitive semantics of our dispute claim language. During the execution of a protocol, the system as a whole goes through a series of well-defined *global states*. The global state consists of the initial secrets (e.g., private keys) of all the players involved, all message exchanges up to that point, and all sources of randomness. Therefore it has enough information to assign values to all the variables that can possibly appear in a dispute claim phrased in our claim language. Given a dispute claim phrased in our claim language, a verifier who knows the entire global state can decide with certainty if the claim is true or not. However, it is extremely unlikely that any verifier can know the entire global state (e.g., private keys of other players). The goal of dispute resolution is for the verifier to attempt to partially reconstruct the *sequence of global states* (along with as much of their contents as possible or necessary) the system has gone through, arriving at the *current state*. A verifier can do this using the *evidence* presented to it. Based on interpretation of the evidence (which is a payment system-specific function), the verifier can assign values to *some* of the variables. Such a *partial assignment* may be contingent on the trustworthiness of the entities involved in the creation of the evidence. The claim is evaluated with respect to the current state, and/or the sequence of states traversed so far.

Once a sufficiently complete interpretation of a state is available, the verifier can determine if a given basic_stmt $s$ is true in that state. The meanings of

| | | |
|---|---|---|
| claim | ::= | role **claims** claim_stmt |
| claim_stmt | ::= | modal_stmt ‖ ¬ modal_stmt ‖ (modal_stmt) modal_stmt ∧ modal_stmt |
| modal_stmt | ::= | possibility_stmt ‖ certainty_stmt ‖ basic_stmt |
| certainty_stmt | ::= | **always** basic_stmt ‖ **never** basic_stmt |
| possibility_stmt | ::= | role_set **could_without** role_set basic_stmt ‖ role_set **can_without** role_set basic_stmt ‖ **once** basic_stmt |
| role_set | ::= | role ‖ role, role_set |
| basic_stmt | ::= | role_part ∧ attr_part |
| role_part | ::= | pt=PAYMENT ∧ payer=*id_val* ∧ payee=*id_val* ‖ pt=VALUE_CLAIM ∧ user=*id_val* ∧ bank=*id_val* ‖ pt=VALUE_SUBTRACTION ∧ user=*id_val* ∧ bank=*id_val* |
| attr_part | ::= | true‖ attr_val_pair ∧ attr_part |
| attr_val_pair | ::= | amount relop *amount_val* ‖ time relop *time_val* ‖ ref=*ref_val* |
| relop | ::= | $<$ ‖ $\leq$ ‖ $=$ ‖ $\neq$ ‖ $\geq$ ‖ $>$ |
| role | ::= | payer ‖ payee ‖ user ‖ bank |

Table 2: Grammar for the Payment Dispute Claim Language

the modal operators are intuitive. The statement "**always** $s$" is true at a state $S$ if $s$ is true in $S$ as well as in every state reachable from $S$. The meaning of "**never** $s$" is analogous. The statement "P **can_without** Q $s$" is true in $S$ if there is a state $S'$ where $s$ is true, *and* it is possible for P to cause the transition from $S$ to $S'$ without any action from Q. Given a path $p = \{S_0, \ldots S_n\}$, the statement "P **could_without** Q $s$" is true in $p$ if the following two conditions are satisfied: (a) "P **can_without** Q $s$" is true at some state $S$ in $p$, and (b) if, at some later state in $p$, it was no longer possible to reach a state where $s$ is true, then P was responsible for this change. Given $p = \{S_0, \ldots S_n\}$, "**once** $s$" is true in $p$ if $s$ was true in a state $S$ in $p$.

Now, let us try to define the semantics more formally. Our model consists of a set of *states* $S$, a set of *roles* $R$, a set of *transitions* $T \subseteq S \times S$, and an interpretation $L$ which assigns a value of the appropriate type to variables in the language.

A role uniquely identifies a service access point (e.g., payer). We assume that there is an infrastructure that enables a verifier to unambiguously identify and authenticate the identity of a player (which is some value from the domain *id_val*) playing any given role. We assume that a multi-party protocol can be described by a directed acyclic graph (DAG), the edges of which correspond to message transmissions between the players, and the nodes correspond to internal global states[4]. We can capture this property by defining that each state has a cardinality, defined by the function $card() : S \rightarrow N$, where $N$ is the set of natural numbers. The cardinality is used to impose a partial temporal order over $S$. If $(S_1, S_2) \in T$, then $card(S_2) > card(S_1)$. If the protocol is specified in the form of local finite state machines, they can first be unwound into a set of DAGs which can then be combined into a single DAG. Replays of protocol messages, when detected and rejected by the receiver, do not influence the global state and therefore do not affect the DAG. If a protocol allows different messages with the same message type to be sent several times during a protocol execution, the different occurrences are represented as seperate transitions in the DAG.

The sender of a message is the agent associated with the edge that represents the message in the DAG. A function $agent() : T \rightarrow R$ identifies the role associated with a given transition. (Recall that an interpretation will associate an *id_val* with a role, thereby associating an *id_val* with each transition as well.) Given a *path* $p$, in the form of a sequence of states $\{S_0, S_1, \ldots, S_n\}$, the function $agents(p)$ returns the union of $agent(S_i, S_{i+1})$, for $i = 0 \ldots n-1$.

The verifier has a payment system-specific evaluation function which can be used to associate a par-

[4] The DAG representation, rather than representing exact protocol states along the different possible execution paths, should be seen as a "template" allowing the verifier to map pieces of evidence to *idealized* states as defined by the protocol.

tial assignment with a given state. The relational operators have the usual semantics. Thus, given a sufficiently complete partial assignment, and a proposition involving an attribute, a relational operator, and a value, it is possible to evaluate if the proposition is true. Given a global state $S$ with a partial assignment, and a basic_stmt $s$ of the form "role_part attr_part" (where role_part and attr_part are conjunctions of propositions as described in the previous section), $s$ is true in $S$, if role_part and attr_part evaluate to true after the partial assignment is made. Since the assignment is partial, the verifier may not always be able to decide whether a claim is true or not, for example, if the evidence supplied is incomplete.

Figure 3 illustrates the semantics of the modal operators. The figure shows the DAG description of a protocol, including the states where a certain basic_stmt $s$ is true.

1. If the statement '**always** $s$' is true in a state (e.g., $S_{100}$), then $s$ (as well as '**always** $s$') is true in that state and in all states reachable from it, in all possible paths. Similarly, if '**never** $s$' is true in a state (e.g., $S_3, S_{201}$), then '**not** $s$' (as well as '**never** $s$') is true in that state and in all states reachable from it.

2. The statement 'P$_2$ **can_without** P$_1$ $s$' is true in $S_1$ because P$_2$ can cause the transfer to $S_{100}$.

3. The statement 'P$_2$ **could_without** P$_1$ $s$' is true in the path $\{S_0, S_1\}$ because of 2. It is also true in the path $\{S_0, S_1, S_{110}\}$ even though $s$ itself cannot be true in $S_{110}$ or any state reachable from $S_{110}$. This is because, in $S_1$ it was still possible to reach a state where $s$ would have been true ($S_{100}$) and it was P$_2$ which chose not to effect that transition.

4. Given a path, the statement '**once** $s$' has the usual meaning in linear temporal logic — for example, '**once** $s$' is true in $\{S_0, S_1, S_2, S_{200}\}$ and $\{S_0, S_1, S_2, S_{200}, S_{201}\}$.

We now define the semantics of modal operators more formally. We first define a valid path as:

```
For a path $p = \{S_0, S_1, \ldots S_n\}$,
    $valid\_path(p)$ iff
        $(S_i, S_{i+1}) \in T,\ i = 0 \ldots n-1$
```

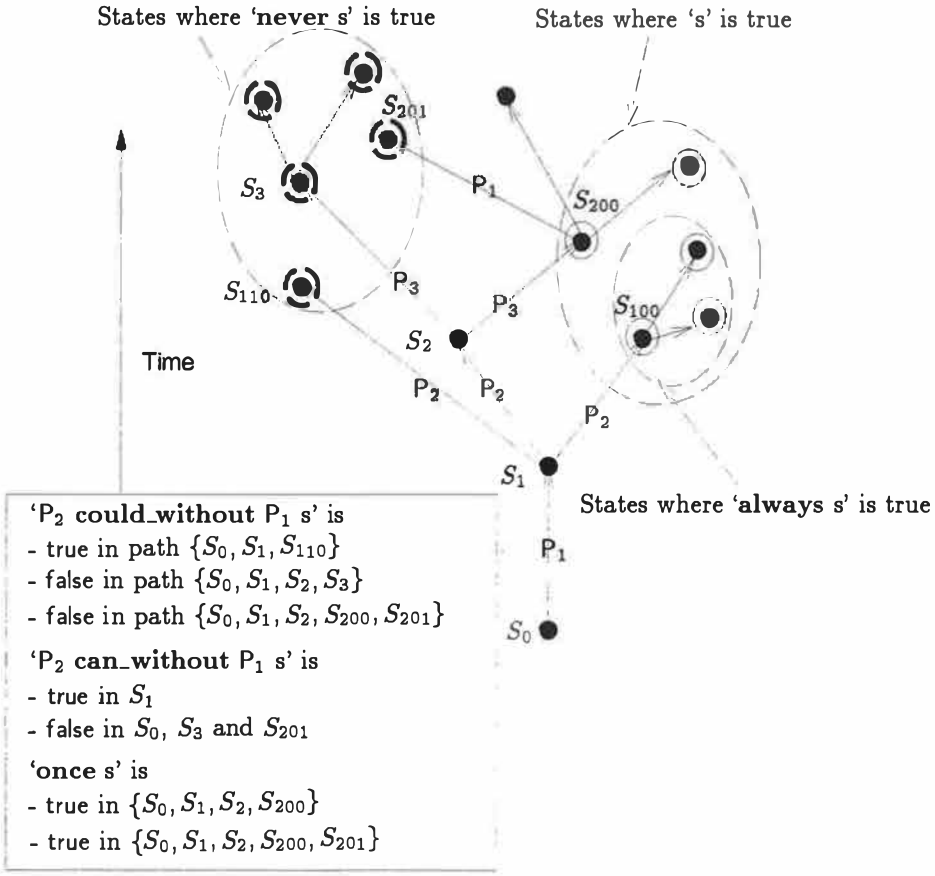
In the following definitions, paths are implicitly assumed to be valid paths. The semantics of the **can_without** operator are now defined as follows:

```
$S \vdash PSET$ can_without $QSET\ s$, iff
    $\exists\ S_n$ such that
        $S_n \vdash s$
    $\wedge\ \exists\ p = \{S_0 = S, S_1, \ldots S_n\}$ such that
            $agent(S, S_1) \in PSET$
        $\wedge\ QSET \cap agents(p) = \phi$
```

That is, given a state $S$, if there is a valid path $p$ leading to a state $S_n$, such that $s$ is true in $S_n$, and the first transition in $p$ can be made by a member of $PSET$ and no one from $QSET$ is required to make any transition in $p$, then the statement '$PSET$ **can_without** $QSET$ $s$' is true in $S$. The **can_without** operator is used to make a statement about the possible future states of the system. In contrast, the **could_without** operator is more general. It is defined with respect to a path (more precisely, with respect to a state and a *specific* path leading to that state). It can be used to make a statement about the system at the end of the path about where it *can* go in the future, as well as where it *could have* gone in the past. The semantics of the **could_without** operator can be defined in terms of the **can_without** operator:

```
For a path $p = \{S_0, S_1, \ldots S_n = S\}$,
    $p \vdash PSET$ could_without $QSET\ s$, iff
        $S \vdash PSET$ can_without $QSET\ s$
    $\vee\ \exists\ S_i \in p, i = 0 \ldots n-1$ such that
            $S_i \vdash s$
        $\wedge\ S_{i+1} \vdash \neg\ s$
        $\wedge\ agent(S_i, S_{i+1}) \in PSET$
    $\vee\ \exists\ S_i \in p, i = 0 \ldots n-1$ such that
            $S_i \vdash PSET$ can_without $QSET\ s$
        $\wedge\ \exists\ S_j \in p,\ j = i \ldots n-1$ such that
                $S_j \vdash ALL$ can_without $QSET\ s$
            $\wedge\ S_{j+1} \vdash \neg\ ALL$ can_without $QSET\ s$
            $\wedge\ agent(S_j, S_{j+1}) \in\ PSET$
```

The set $ALL$ represents the set of all roles for the primitive transaction referred to in $s$. The rule identifies three disjunctive conditions to evaluate the truth of the statement '$s_{could}$=$PSET$ **could_without** $QSET$ $s$' with respect to a path $p$. The first disjunction says $s_{could}$ is true if '$s_{can}$=$PSET$ **can_without** $QSET$ $s$' is true in the last state of $p$. The second disjunction says that



'$P_2$ **could_without** $P_1$ s' is
- true in path $\{S_0, S_1, S_{110}\}$
- false in path $\{S_0, S_1, S_2, S_3\}$
- false in path $\{S_0, S_1, S_2, S_{200}, S_{201}\}$

'$P_2$ **can_without** $P_1$ s' is
- true in $S_1$
- false in $S_0$, $S_3$ and $S_{201}$

'**once** s' is
- true in $\{S_0, S_1, S_2, S_{200}\}$
- true in $\{S_0, S_1, S_2, S_{200}, S_{201}\}$

Figure 3: Semantics of Dispute Statements

if $s$ was true at some state in $p$, but not at the state immediately following it, *and* the transition was caused by someone in *PSET*, then $s_{could}$ is true with respect to $p$. The third disjunction is a little more complicated. The intent is to capture the following case. Sometime in the past, it was possible to reach a state where $s$ is true (i.e., there was a path, say $p'$ from some state $S_i$ in $p$, making $s_{can}$ true at $S_i$). The agents of $p'$ consist of some members of *PSET* but none from *QSET*. It may also consist of *other* entities, not in either of the above sets. The statement '$(PSET \cup agents(p'))$ **can_without** $s$' should therefore hold at every state in $p$ subsequent to $S_j$. If it fails to hold after a certain transition, *and* that transition was caused by a member of *PSET*, then we can assert that $s_{could}$ is true in $p$. In the interest of simplicity, we relax the definition a little by using the set of all roles (*ALL*) instead of $(PSET \cup agents(p'))$. The last disjunction captures this property. Similarly,

$p \vdash$ **once** $s$, iff $\exists\ S \in p$ such that $S \vdash s$

$S_0 \vdash$ **always** $s$, iff $\forall\ p = \{S_0, \ldots\},\ p \vdash \Box\ s$

$S_0 \vdash$ **never** $s$, iff $\forall\ p = \{S_0, \ldots\},\ p \vdash \Box \neg\ s$

The notation "$p \vdash \Box\ s$" has the usual meaning in linear temporal logic: in the sequence of states $p$, the formula $s$ holds true in every state. In Section 3.3, we will look at an example payment system to see how we can build a global state transition diagram.

The claim language described above constitutes the set of symbols and grammar rules necessary for *specifying* dispute claims in the generic payment service. The language constructs allow multiple time-lines. Therefore, it belongs to the family of branching temporal logics [7]. It does not include all possible temporal logic constructs: for example, the **until** operator. We have only included the constructs that seemed necessary to express the claims described earlier. However, we have included constructs like **can_without** and **could_without**, which are not standard branching temporal logic constructs.

The language can be extended as needed. In Section 3.2 below, we describe how the claim language can be extended to derive the language describing the messages in a generic dispute protocol between the verifier and the player.

The inference mechanisms used by the verifier constitute a logic over the claim language described. Proving soundness and completeness of this logic with respect to a given payment system actually means proving the payment system correct. Our approach has rather been to add dispute handling to existing payment systems, most of which do not fulfill these correctness requirements. Therefore, we consider the main contribution of this work to be in the problem definition and the development of the claim language; not in the development of the logic.

### 2.3.3 Examples

The five dispute claims mentioned in Section 2.1 correspond to the following statements in our language:

- PAYMENT payer=Alice payee=BobAir amount=$200
- PAYMENT payer=Alice payee=BobAir amount=$200 time < Jan12
- ¬ PAYMENT payer=Alice payee=BobAir amount=$200
- payee **could_without** payer PAYMENT payer=Alice payee=BobAir amount=$200 time < Jan12
- ¬ VALUE_SUBTRACTION *user*=Alice *bank*=CarolBank amount=$300 ref ="#434: for flight 822 on Jan 19:payment to BobAir"

If players have evidence proving that a certain value transfer transaction has reached a guaranteed final state, they may choose to make stronger claims, using the **always** or **never** operators. For example, if Alice has a receipt for the payment made using a payment system which does not support refunds, she may claim "**always** PAYMENT payer=Alice payee=BobAir amount=200" instead.

# 3 Supporting Claims with Evidence

## 3.1 Architecture for Dispute Handling

### 3.1.1 Overview

Recall that there are three types of players in a dispute: the *initiator* who starts the dispute by making a claim, the *verifier* who co-ordinates the dispute handling and possibly makes, or helps make, the final decision about the validity of the claim, and a set of *responders* who may be asked by the verifier to participate in the process.

At the access point of each player, there is a "user" part (which may be the human user, or an application program acting on his behalf) and a "system" part (which is an implementation of the dispute service: e.g., an iKP implementation of the generic payment dispute service). The verifier's system has two parts: the *inference engine* receives a claim, and associated non-repudiation tokens, analyses them, and determines the conditions under which the claim is true, and the conditions under which it is false; the *policy engine* uses the result of the first part to make a final decision. The policy engine may be a human arbiter, external to the dispute handling service. It needs to use trust assumptions. In the simplest case, this may be in the form of a blacklist of untrusted principals. It is possible that the trust assumptions require a more elaborate representation (for example, as in PolicyMaker [4]).

Each player's system then engages in a proof protocol with the verifier's system. Recall that during the dispute protocol, the verifier's goal is to determine both the current state of the payment system, and the sequence of states through which it has progressed, and that the validity of the claim should be evaluated with respect to this state and sequence. At the end of the protocol, the verifier's system returns an analysis of the claim. The analysis consists of a likely decision (yes or no) as well as the set of players who were witnesses to the decision and the set who were against it. If there is insufficient evidence, the verifier's system may throw an exception. The policy engine makes the final decision by combining this analysis with his trust assumptions.

This leads to the following requirements on the design:

- The verifier needs the following service primitives:
  - map: Takes a *claim* as input and returns

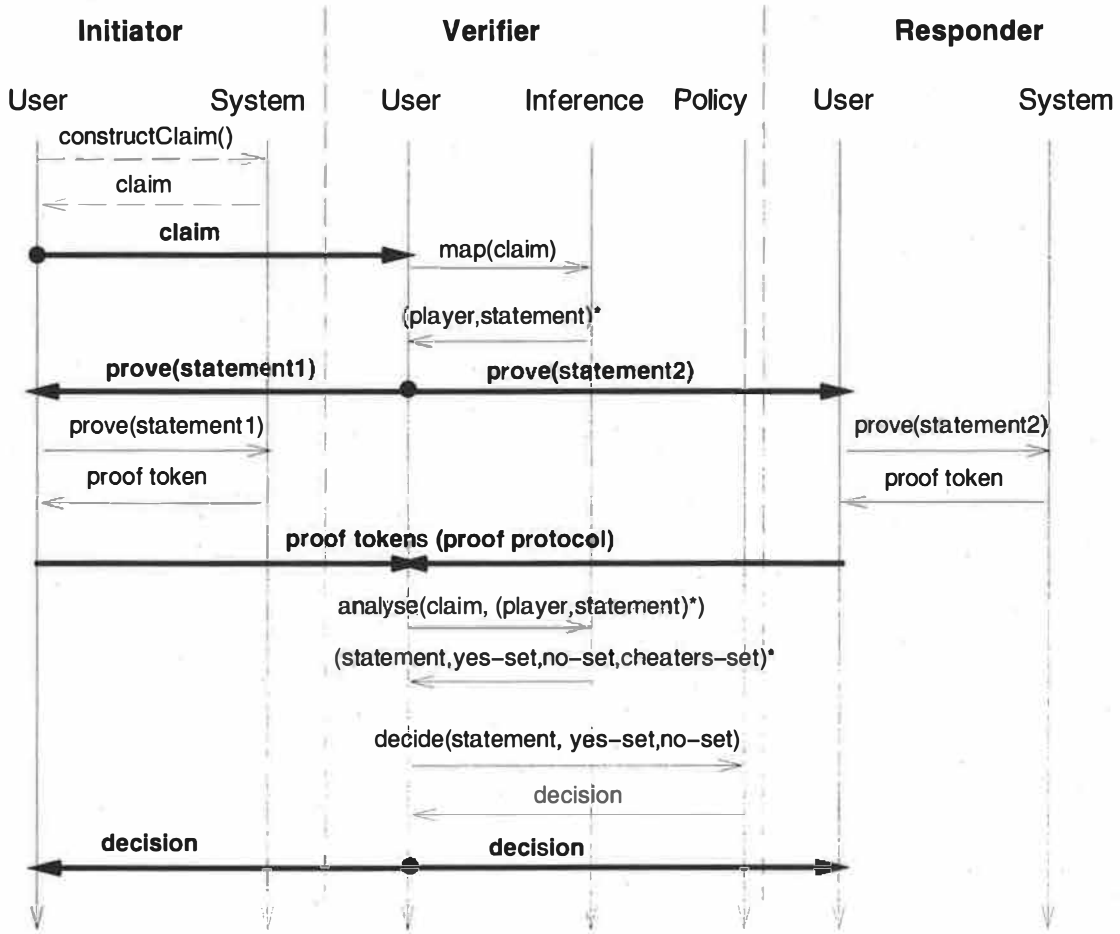


Figure 4: Basic Dispute Protocol

a list of (*player*, *statement*) pairs. Each player is required to prove the corresponding statement.

- analyse: Takes a *claim*, a set of players, and the statements they need to prove as input, engages in some proof protocol (maybe as simple as receiving one or more non-repudiation tokens, interpreting them, and making an inference on the statements proved), and returns the set of players according to which the claim is true, the set of players according to which it is false, and the set of players who have been found to be cheating.
- decide: Takes a *claim*, and two sets of players (*yes*-set, *no*-set) as input, and returns one of yes, no, or cannot-decide, based on the verifier's trust relationships as output. This is the result of the dispute.

- Each player needs the following primitives:
  - constructClaim: which allows the user to construct a claim, and
  - prove: which takes a statement as input and attempts to prove it (maybe as simple as retrieving the pieces of evidence and returning them to the caller).

A concrete design of a dispute service for the generic payment service of [1] is described in [2].

### 3.1.2 Lost and withheld evidence

In some scenarios, a claimant may need to rely on another party to help prove a certain claim. This may be because the claimant lost parts of the necessary evidence; it may also be an inherent property of the payment system; e.g., in a payment system where the payer never gets signed receipts from the payee, the payer may not be able to win a payment dispute without cooperation from his bank. In either case, the other party could decide to withhold the necessary evidence. This problem cannot be dealt with except in the case where it can be shown that the other party should indeed possess or have possessed the evidence. It is up to the specific payment system to define the actions to be taken in such a case.

### 3.1.3 Enhancements

We have limited ourselves to disputes in a generic payment service where there is only one service boundary. The system is "below" the boundary and the user or his application is "above" the boundary. Comprehensive electronic commerce frameworks such as SEMPER [17] are structured into multiple layers. A payment usually takes place in the context of a higher layer transaction (e.g., a fair exchange) which in turn may take place in the context of a transaction in the layer above (e.g., an instance of an on-line purchase application). A dispute claim made in a higher layer needs to be suitably mapped to corresponding claims in the lower layers. The running of the dispute protocol needs to be co-ordinated among the different levels. This is left as an open problem.

## 3.2 Evidence and Trust

During a dispute, players have to support dispute claims by proving certain statements to the verifier. The ability to prove statements comes from pieces of evidence (evidence tokens) accumulated during a transaction of the primary service. The simplest form of evidence is a non-repudiation token that can be verified by anyone who has the necessary public keys, certificates, certificate revocation lists etc. The proof protocol in this case simply consists of presenting the token to the verifier. There can be more involved proof protocols, such as in "undeniable signature schemes" [5]. In the following, we will assume only simple proof protocols consisting of the presentation of non-repudiation tokens.

Often, non-repudiation tokens cannot substantiate an absolute statement. In general, an evidence token corresponds to a non-repudiable assertion by one or more players that they believed the protocol reached a certain state (with an associated partial assignment). Such an assertion is a *witnessed statement*. During the dispute protocol, the verifier asks various players to prove witnessed statements (in the proof request messages in the dispute protocol of Figure 4). We define the language for these witnessed statements by building on our claim language in Section 2.3 and extending it with the following rule.

```
witnessed_stmt ::= role witnessed asserted_stmt
asserted_stmt  ::= basic_stmt ‖ certainty_stmt
```

The **witnessed** operator takes two parameters: an asserted_stmt *s*, and a role *P*. The statement "P **witnessed** *s*" is true if P has non-repudiably asserted that the transaction reached a certain state, with an associated partial assignment in which *s* evaluates to true. The ability to prove and verify such a non-repudiable assertion itself assumes a lower layer dispute service for non-repudiation.

The verifier's analyse method, after having collected the non-repudiation tokens proving the witnessed statements, returns a result of the form:

> ∨{[¬] <claim_stmt> yes={...}, no={...}}, cheating={...}

Each component of the disjunction contains either the original statement in the claim or its negation, and two optional sets of players: the set of players whose statements are in agreement with the conclusion and the set of players whose statements are contrary to the conclusion. The verifier will pick one of these disjunctions, depending on the set of players he trusts. If the evidence presented is insufficient to decide one way or the other, the system may raise an exception. Further, it may also be able to detect if some player has cheated (for example, if the same player has witnessed a certain statement and its exact opposite, it may imply that the player had cheated).

Note that a carefully designed, secure, payment system can be rendered insecure if the inference engine used by the verifier is wrong. It is important to make sure that the inference engine used not degrade the security of the protocol.

Consider as an example a dispute claim discussed earlier: Alice claims to have made a payment of $200 to BobAir. There is no way to say with absolute certainty whether the transaction actually took place. A receipt from BobAir proves the statement

BobAir **witnessed** PAYMENT payer=Alice payee=BobAir amount=$200

Whether a verifier can infer

PAYMENT payer=Alice payee=BobAir amount=$200

depends on the context, the trust assumptions of the verifier, and even on the actual payment system allegedly used for the value transfer. For example, in a dispute against BobAir, the verifier could accept a signed receipt from BobAir as sufficient evidence to conclude that the latter claim is true. However, if BobAir and Alice are in collusion and want to convince a third party (say the tax authorities) that a certain payment happened, BobAir's signed receipt alone is not sufficient. In this case, if the verifier trusts the bank, it can accept a signed statement from the bank as sufficient evidence. Thus an analyse result of the form:

PAYMENT <role_part> <attr_part> yes={bank} no={}

may allow the verifier to reach a final positive decision, whereas a result of the form:

PAYMENT <role_part> <attr_part> yes={payer,payee} no={}

cannot exclude collusion of payer and payee in trying to prove a payment.

Now, in Section 3.3, we will look at a simplified version of the iKP protocol to see (a) how statements can be supported with evidence and (b) how to derive system specific inference rules.

## 3.3 An Example: Evidence Tokens in iKP

iKP was designed as a solution for securing credit card payments over open networks. The original protocol was described in [3]. A more detailed definition with some improvements is available in [15]. In this section, we present a simplified version of the 3-party iKP (3KP) where all three players are assumed to have signature and encryption key pairs. We will see how the receipts gathered during an iKP protocol run can be mapped to dispute statements defined in Section 2.3 for the generic payment service.

### 3.3.1 Protocol Description

In our simplified version of iKP, there are three players: Customer (Payer), Merchant (Payee), and Acquirer (Bank). Before the transaction begins, the customer and merchant agree about the amount of payment ("price") and the description ("desc") of what the payment is for. The first half of Table 3 depicts the initial information of each player. To begin the transaction, the payee:

- generates two random nonces v and vc; these nonces will later be used as part of the receipt(s) from the payee,
- collects the common information ("com"). The common information consists of all the pieces of information that will be known to all the parties at the end of the transaction,[5] and
- generates a signature $\text{Sig}_M$ containing hashes of the data items mentioned above and sends the signature along with any necessary information to the payer. This is the signed *offer* from the merchant.

The signature on the offer makes it non-repudiable. But this is not relevant to our discussion.

The payer then sends an *order* message. This is the authorisation by the customer to make the payment. The order is a signature $\text{Sig}_C$ of the payer on two pieces of information,

- $\mathcal{H}(com)$, and
- an encryption of the price, the customer's account number ("CHI"), and $\mathcal{H}(com)$.[6]

The payee forwards the *order* along with the *offer* to the acquirer requesting authorisation. The acquirer replies indicating whether the authorisation succeeded or not. For simplicity, let us assume that the acquirer immediately transfers the money from payer to payee if the authorisation is successful. The authorisation response $\text{Sig}_A$ from the acquirer is signed.

[5] One item in the common information is a randomised hash of the description ($\mathcal{H}^R$(desc)). The payer and payee already know the description. The randomised hash allows the bank to confirm that the payer and payee agree on the description without the bank's having to know the actual text of the description

[6] Parts of the card-holder information is considered "secret" information (e.g., like a credit card number). The encryption in the *order* can be opened only by the bank — thus, the payee will not be able to determine CHI.

| Data | | Functions | |
|---|---|---|---|
| $M$ | Merchant (payee) | $\mathcal{H}()$ | A one-way hash function |
| $C$ | Customer (payer) | $\mathcal{H}^R(m)$ | A randomised one-way hash of $m$ |
| $A$ | Acquirer (bank) | $\mathcal{E}_X^R(m)$ | A randomised encryption |
| TID | Transaction ID | | of $m$ with the public key of $X$ |
| date | Time stamp | $\mathcal{S}_X(m)$ | Signature of $X$ on $m$ |
| price | Amount information | | |
| desc | Description of payment | | |
| v | Confirmation Authenticator | | |
| vc | Cancellation Authenticator | | |
| CHI | Card-holder Information | | |
| com | Common information $(M, C, \text{price}, \text{TID}, \text{date}, \mathcal{H}^R(\text{desc}), \mathcal{H}(\text{v}))$ | | |

**Payment**

$$\textit{offer} : C \xleftarrow{\text{TID}, \text{date}, \mathcal{H}(\text{v}), \text{Sig}_M = \mathcal{S}_M(\mathcal{H}(com), \mathcal{H}(\text{v}), \mathcal{H}(\text{vc}))} M$$

$$\textit{order} : C \xrightarrow{\text{Enc} = \mathcal{E}_A^R(\text{price}, \mathcal{H}(com), \text{CHI}), \text{Sig}_C = \mathcal{S}_C(\text{Enc}, \mathcal{H}(com))} M$$

$$\textit{auth-request} : M \xrightarrow{\text{TID}, \text{date}, \mathcal{H}(\text{v}), \mathcal{H}(com), \mathcal{H}^R(\text{desc}), \text{Enc}, \text{Sig}_M, \text{Sig}_C} A$$

$$\textit{auth-response} : M \xleftarrow{\text{yes}||\text{no}, \text{Sig}_A = \mathcal{S}_A(\text{yes}||\text{no}, \mathcal{H}(com))} A$$

$$\textit{confirm} : C \xleftarrow{(\text{v}, \text{yes})||(\text{vc}, \text{no}), \text{Sig}_A} M$$

**Cancellation**

$$\textit{cancel} : C \xleftarrow{\text{vc}} M$$

**Refund**

$$\textit{refund-request} : M \xrightarrow{\text{Sig}_M^{refund} = \mathcal{S}_M(\mathcal{H}(com), \text{price}, \text{“refund”})} A$$

$$\textit{refund-response} : M \xleftarrow{\text{Sig}_A^{refund} = \mathcal{S}_A(\mathcal{H}(com), \text{price}, \text{“refund”})} A$$

Figure 5: Simplified iKP Protocol

| Initial Information | |
|---|---|
| payer | desc, price, CHI |
| payee | desc, price, TID, date, v, vc |
| bank | |
| **Collected Information** | |
| payer | com, $\mathcal{H}(\text{v})$, $\text{Sig}_M$, $[\text{Sig}_A]$, v$\|$vc |
| payee | com, Enc, $\text{Sig}_C$, $[\text{Sig}_A]$, $[\text{Sig}_A^{refund}]$ |
| bank | [com, Enc, CHI, $\text{Sig}_M$, $[\text{Sig}_M^{refund}]$, $\text{Sig}_C$] |

Table 3: Information of Players in a Completed iKP Transaction

| Role | Evidence | Statement |  | Possible Counter |
|---|---|---|---|---|
|  |  | witness | Primitive Transaction |  |
| payer | 1. $\mathrm{Sig}_M$, v, com | payee | PAYMENT | 8 |
|  | 2. $\mathrm{Sig}_A$, yes, com | bank | VALUE_SUBTRACTION | 10 |
|  | 3. $\mathrm{Sig}_A$, no, com | bank | **never** VALUE_SUBTRACTION |  |
|  | 4. $\mathrm{Sig}_M$, vc, com | payee | **never** PAYMENT | 11 |
| payee | 5. $\mathrm{Sig}_C$, com, Enc | payer | PAYMENT | 3, 4, 10 |
|  | 6. $\mathrm{Sig}_A$, yes, com | bank | VALUE_CLAIM | 10 |
|  | 7. $\mathrm{Sig}_A$, no, com | bank | **never** VALUE_CLAIM |  |
|  | 8. $\mathrm{Sig}_A^{refund}$, com | bank | **never** VALUE_CLAIM |  |
| bank | 9. $\mathrm{Sig}_M$, $\mathrm{Sig}_C$, com | payee | VALUE_CLAIM | 8 |
|  | 10. $\mathrm{Sig}_M^{refund}$, com | payee | **never** VALUE_CLAIM |  |
|  | 11. $\mathrm{Sig}_C$, com, CHI | payer | VALUE_SUBTRACTION | 3, 4 |

Table 4: Mapping Evidence to Dispute Statements in iKP

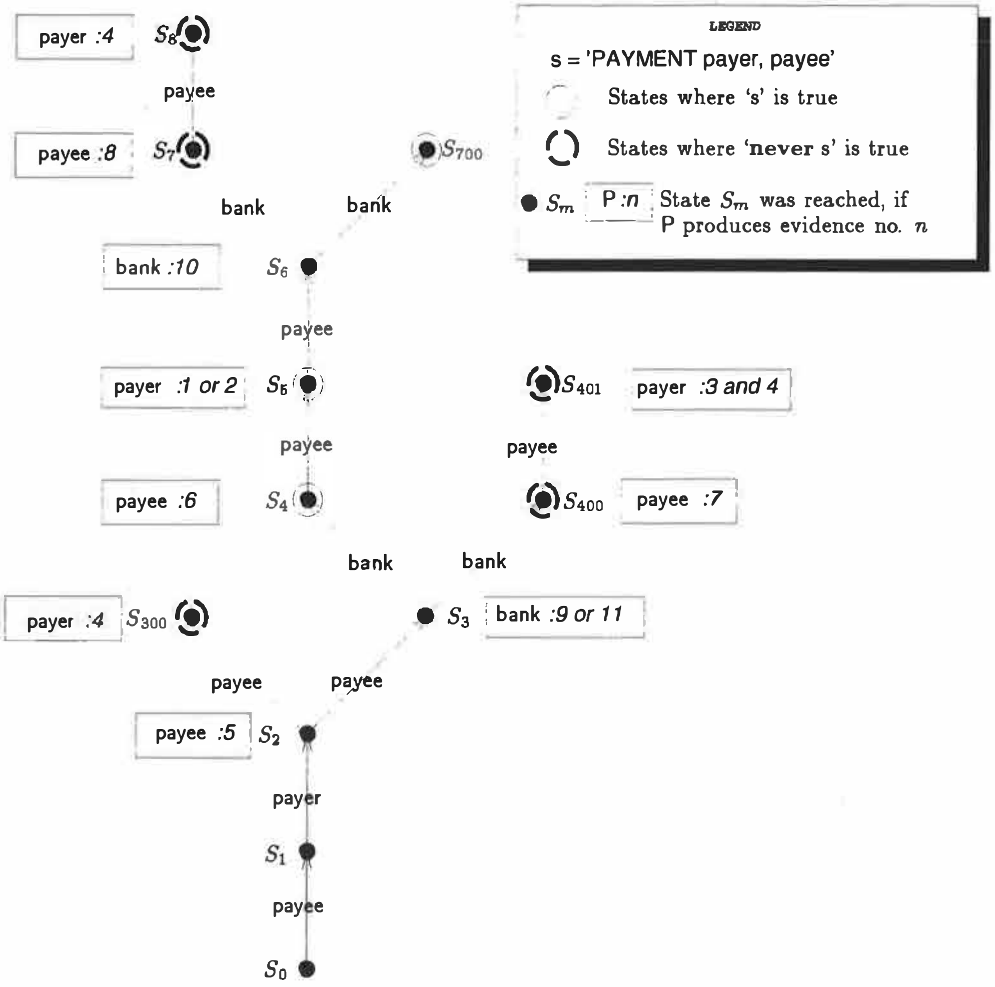


Figure 6: Global States in iKP

Once the payee receives the authorisation response, he will send a confirmation to the payer. The confirmation contains the acquirer's authorisation response and the authenticator v. The pair ($\mathrm{Sig}_M$, v) constitutes a receipt by the payee that he received the payment. If a payee decides to cancel a payment, he can issue a cancellation receipt to the payer by sending him vc. If the payer possesses the pair ($\mathrm{Sig}_M$,vc), it is proof that the payee agreed to cancel the payment. If the payee cancels an already authorised payment, he can contact the acquirer and arrange for a refund.

The second half of Table 3 lists the pieces of information that are collected by the players at the end of a successful protocol run. Again, items within square parentheses are available only under certain circumstances.

Note that instead of using v and vc, the *confirm* and *cancel* flows from the payee to payer can be signed by the payee. The use of v and vc avoids the payee having to make two or more signatures by allowing the original signature to be "extended."

### 3.3.2 Mapping iKP receipts to Dispute Statements

Table 3 lists the pieces of information known to each player at the end of a successful transaction. We can now try to extract evidence tokens from these pieces and identify the dispute claims they can support.

In the actual iKP protocol, the acquirer transfers the money from the customer to merchant during a "capture" transaction. The merchant can also capture a different (lower) amount than was previously authorised. The refund transaction is essentially a negative capture. Depending on the policies of the players, some of the receipts may be omitted. All these variations are not relevant to our discussion. Therefore they are left out from our simplified version.

With the information in Table 4 and Figure 5, we can represent the global states in a run of the iKP payment protocol in the form of a DAG as in Figure 6.

In iKP, all three primitive transactions are completed at the same time. The states where the primitive transactions are succesfully completed are marked with a circle. The states where '**never** *s*' is true (*s* is any basic_stmt with any of the primitive transactions e.g. 'PAYMENT *payer, payee*') are marked with a thick broken circle. Notice how the verifier can use this graph to implement the analyse method (Section 3.1): while 'PAYMENT *rest_of_the_claim*' is false in $S_{300}$, $S_{400}$, and $S_7$, the statement 'payee **could_without** payer PAYMENT *rest_of_the_claim*' is true in $S_{300}$ and $S_7$ (but not in $S_{400}$).

## 4 Summary and Conclusion

We have shown why a generic dispute service is needed for payment systems. We developed a language to express dispute claims and applied it to an example payment system. Finally, we described a generic dispute handling protocol in the context of an architecture for dispute handling. A crucial part of the dispute handling framework is the verifier's inference engine. When a new payment system is adapted to the framework, the critical step is to identify the inference rules applicable to that system. In general, the process of deriving the inference rules is equivalent to proving the payment system correct. Ideally, derivation of inference rules should be an integral part of the design process of the payment system. On the other hand, the aim is not complete automation of dispute resolution. Thus, even with an incomplete set of inference rules, a payment system can be incorporated into the framework. The most trivial inference rule is "ask the human user!"

## 5 Acknowledgements

The authors wish to thank Michael Waidner, Birgit Pfitzmann and Mehdi Nassehi for useful comments and discussions, and the anonymous referees for their helpful suggestions.

## 6 Availability

More information on this project can be found at

`http://www.zurich.ibm.com/Technology/Security/extern/disputes.html`

## References

[1] J. L. Abad-Peiro, N. Asokan, M. Steiner, and M. Waidner. Designing a generic payment service. *IBM Systems Journal*, 37(1):72–88, Jan. 1998.

[2] N. Asokan, E. V. Herreweghen, and M. Steiner. Towards a framework for handling disputes in

payment systems. Research Report RZ 2996, IBM Research, Mar. 1998.

[3] M. Bellare, J. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, and M. Waidner. iKP – a family of secure electronic payment protocols. In *First USENIX Workshop on Electronic Commerce* [16], pages 89–106.

[4] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Research in Security and Privacy, Oakland, CA, May 1996. IEEE Computer Society,Technical Committee on Security and Privacy, The Institute of Electrical and Electronics Engineers IEEE Inc.: Computer Society Press.

[5] D. Chaum and H. van Antwerpen. Undeniable signatures. In G. Brassard, editor, *Advances in Cryptology – CRYPTO '89*, number 435 in Lecture Notes in Computer Science, Santa Barbara, CA, USA, Aug. 1989. Springer-Verlag, Berlin Germany.

[6] B. Cox, J. D. Tygar, and M. Sirbu. NetBill security and transaction protocol. In *First USENIX Workshop on Electronic Commerce* [16].

[7] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*, chapter 16, pages 995–1072. Elsevier Science Publishers B.V., 1990.

[8] M. E. Katsch. Dispute resolution in cyberspace. In *Connecticut Law Review Symposium: Legal Regulation of the Internet*, number 953 in 28, 1996. Available from `http://www.umass.edu/legal/articles/uconn.html`.

[9] P. F. Linington. Fundamentals of the layer service definitions and protocol specifications. *Proceedings of the IEEE*, 71(12):1341–1345, Dec. 1983.

[10] Mastercard and Visa. *SET Secure Electronic Transactions Protocol*, version 1.0 edition, May 1997. Book One: Business Specifications, Book Two: Technical Specification, Book Three: Formal Protocol Definition. Available from http://www.mastercard.com/set/#down.

[11] B. Pfitzmann and M. Waidner. Integrity properties of payment systems, Dec. 1996. Private Communication of work in progress; contact the authors for status of the work.

[12] B. Pfitzmann and M. Waidner. Properties of payment systems - general definition sketch and classification. Research Report RZ 2823 (#90126), IBM Research, May 1996. Submitted for Publication.

[13] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *J. ACM*, 21(2):120–126, Feb. 1978. US Patent 4,405,829: Cryptographic Communications System and Method, Public Key Partners PKP.

[14] Sun Microsystems. The java wallet (TM) architecture white paper, March 1998. Available from `http://java.sun.com/products/commerce/docs/whitepapers/arch/architecture.ps`.

[15] G. Tsudik. Zürich iKP prototype: Protocol specification document. Technical report, IBM Research, Feb. 1996. IBM Research Report RZ-2792.

[16] USENIX. *First USENIX Workshop on Electronic Commerce*, New York, July 1995.

[17] M. Waidner. Development of a secure electronic marketplace for Europe. In E. Bertino, H. Kurth, G. Martella, and E. Montolivo, editors, *Proceedings of the Fourth European Symposium on Research in Computer Security (ESORICS)*, number 1146 in Lecture Notes in Computer Science, Rome, Italy, Sept. 1996. Springer-Verlag, Berlin Germany. also published in: EDI Forum 9/2 (1996) 98-106.

# Electronic Commerce on the Move

**John du Pré Gauntt**
*107 Sussex Gardens*
*London, W22RU*
*tel/fax +44-171-402-4924*
*email ag112120@dircon.co.uk*

## Abstract

This paper will address some of the business factors and technical developments that are driving electronic commerce in mobile telecommunications. The trend towards mass-market penetration coupled with increased processing power in the handset are evolving mobile networks into one of the primary e-commerce platforms in Europe.

The author will introduce two European pilot programs where the mobile handset—instead of a dedicated PC—is the vehicle for digital transactions. One program known as Smart Access is a service model that uses electronic cash, while a second model in Finland employs intelligent network technology to allow the user to 'Dial-a-Coke' from a vending machine and settle the transaction on their monthly telephone bill.

The next section will introduce an aspect of third generation mobile systems known as the Virtual Home Environment (VHE). The VHE is widely considered to be that part of the network architecture where mobile electronic commerce applications or services could reside. The last section will conclude with a comment about mobile electronic commerce.

## Introduction

While the classic model of a personal computer hooked up to the Internet via a dial-up connection enjoys the lion's share of research, development and investment in electronic commerce applications or services, digital mobile telecommunications networks like GSM and PCN (the European term for PCS) are coming into their own as viable platforms for electronic transactions. This technical capability is joined by mobile telecomms' penetration into the mass market that far outstrips personal computers in both scale and speed.

Consequently, electronic commerce models are having to consider mobile applications in a new light. More than the fabled network computer, digital handsets constitute the most effective implementation of the thin client paradigm of information services and communications delivery.

But technology is only part of the story regarding the future of mobile electronic commerce. Concurrent with technical advance, the mobile telecomms industry is now a force to be reckoned with on the global stage. The competitive need for operators to differentiate is the main driving force for electronic commerce applications for mobile markets. E-commerce along with information and entertainment services are seen as vital for escaping the current 'coverage-quality-price' competitive model that characterized the build-up of second generation digital systems.

This paper will sketch some of the forces that are changing the basic value proposition of the mobile industry and are opening new avenues for mobile electronic commerce services. It will then describe two Europe-based mobile e-commerce service models: the Smart Access program that is being piloted by AT&T-Unisource, and a trial scheme in Finland for using short-code dialing for vending machines.

These systems are important given that they cover the transaction spectrum of electronic commerce, ranging from anonymous digital cash-based systems all the way to pure credit functions that are being enhanced through mobile networks. This paper will then turn to network and service creation concepts being considered for third generation (3G) mobile systems in Europe. 3G mobile systems place e-commerce at the heart of their business proposition. The aim of this paper will be to introduce some trends and possible scenarios for electronic commerce in a non PC-based, non-North American environment for the USENIX community.

## I. Mass market mobile in Europe

A quick glance at mobile penetration figures for the European Union (EU) appears to show nothing but 'good news'. Penetration levels in Europe as a whole exceed 9%-equating to more than 35m subscribers.1 This growth should accelerate for the rest of the decade and should not plateau until at least 2005. Salomon Brothers forecasts 15% annual subscriber growth and over 11% growth in annual revenues, predicting that Europe's mobile penetration will exceed 30% within nine years.2

A 30% overall penetration level would place Europe as a whole on equal terms with today's pace-setting Nordic region. Indeed, estimates for these developed mobile markets suggest that future penetration levels of over 50% are feasible.3 Such penetration would place mobile telecomms roughly in the same category as TVs, radios or other mass-market domestic appliances.

Concurrent with wider overall penetration of mobile is more formal attention being paid by the corporate user market, which has begun to view mobile services as an significant part of its overall telecomms spend.4 Mobile services are being used increasingly to connect the various organizational 'flavors' found within multinationals-i.e. wholly-owned business units, joint-ventures, supplier/customer trading areas etc.

Yet, as mobile technology inserts itself more deeply into the agendas of major industries and public policy, the basic model that built the mobile industry will be forced to change to meet the exigencies of a mass market. One can expect traditional high-margin, subscription-based revenues sources such as handset sales, connection charges and monthly rental fees to decline in favor of larger (but lower margin) network usage-based revenue sources.

As a result, one can see a gradual amalgamation and rationalization of mobile interests by multinational operators which is a further indication that the mobile industry is becoming similar in structure—and subject to some of the same competitive forces—as many other consumer industries. One can argue that as soon as a telecommunications service becomes significantly unbundled from the network infrastructure and starts moving towards an incremental cost-based tariff, one would expect to see a lot of co-operation, consortium formation and consolidation at the regional level.

It is also apparent that these aggregators will seek to take mobile services further up the info-communications value chain by offering—among other things—electronic commerce services. This is in no small part due to the need to create new avenues for traffic growth and customer retention. E-commerce applications open a new operational mode for mobile services, in effect transforming the mobile terminal into a mediating device for navigating among service providers in addition to being a communications instrument.

There are factors aside from the technical features of mobile terminals that lead many to believe that mobile e-commerce is a natural progression for the industry. The back-end functions of mobile networks are thought to lend themselves to mobile e-commerce. Fraud systems and churn applications—software designed to detect the likelihood of a customer switching to a competitor—are linked with customer care applications into an overall billing architecture that can capture and account for multiple small transactions through 'per-second' billing over the wide area. Moreover, this billing information is analyzed and shared with network planning modules in order to provide a quality of service (QoS) for the entire network.

In addition, mobile service providers regularly host visitors on their networks and have a well developed and scaleable architecture for capturing, handling, accounting for and charging back visitor usage on the network. Clearing houses such as the Luxembourg-based Multinational Automated Clearing House (MACH) are able to offer mobile operators the option of managing their typically 75-80 bi-lateral roaming agreements through a single organizational shell which allows members to settle all of their obligations in a single currency. Other clearing houses such as the London and Washington DC-based Cibernet offer similar types of financial services for operators.

In addition to terrestrial operators, clearing houses such as MACH have signed agreements with the nascent Global Mobile Personal Communications Services (GMPCS) operators like Iridium. In this way, satellite communications users will be able to roam across cellular networks with the same ease as their present GSM counterparts. Given the background of organizations like MACH and Cibernet in clearing call detail records (CDRs), the conceptual jump to substitute anonymous payments is rather straight forward. Thus, from a terminal, network infrastructure and business practice point-of-view, one can argue that mobile telecomms networks are further along the evolutionary path for offering electronic commerce services than many of their PC or fixed-connection counterparts.

## II. Mobile e-commerce trials in Europe

The catch phrase for AT&T-Unisource's Smart Access program is "we efficiently sell what the network can deliver". Smart Access is a service concept that enables users to make payments via the Internet by using their smart card with either a PC, a hand-held computer or a mobile phone. Smart Access is based on Internet standards (TCP/IP, HTTP and MIME) and is designed to be integrated with existing hardware and software products or platforms.

Intended for—but not restricted to—transactions of less than US$10 (where credit card systems tend to be uneconomic), Smart Access is positioned to allow digital goods and services to be sold in precise measures. The cash transaction provides the user with the requested digital product or service immediately, without having to go through a subscription process with the information provider. Smart Access can be thought of as a payment service for companies wanting to sell low-value goods over the Internet in a device independent environment.

In order to move towards an open transaction model, Smart Access had to work in a thin client environment and not just on PCs. Another requirement was that there would be no client-side cryptography. Since smart cards handle consumer security, and since the Smart Access architecture allows merchants, catalogue operators and banks to use as much encryption and authentication technology as they want, no cryptography (and therefore no complex processing) is needed at the client end.

The basic merchant/consumer interaction works as follows. A consumer browses a catalogue and makes a selection. When the consumer has finished making selections and wants to buy, the catalogue checks the selection and generates a bill for the selected goods or services. This is called the Internet Payment Ticket (IPT). The consumer and the catalogue then negotiate a payment choice based on a digital cash systems (for example, Proton, Visacash, Mondex etc.), a negotiation that will conclude with the consumer being directed to a payment server. The consumer then sends the IPT to the payment server and pays with the agreed-upon digital cash.

Once the payment server has received the e-cash, it signs the IPT to form a proof-of-payment Digital Receipt (DR). The consumer sends the DR to a content server to get whatever they selected. The content server checks the DR, knows that it has a valid order and that payment has been made and it can start delivery or anything else needed to fulfill the transaction. The catalogue, content and payment servers are logically distinct and could be physically distinct as well.

So long as the content server gets a DR, it does not care how payment was made. In theory, this means that not only is the merchant able to accept many more payment mechanisms, the merchant may not even know which payment system was used. In this model, the retailer has outsourced the entire payment process to the banking sector.

Among the financial service providers that have partnered with AT&T-Unisource in exploiting Smart Access is Banksys, a subsidiary of Belgium's banks that is the national EFT/POS operator and authorization center. Banksys developed the Proton smart card scheme which it has subsequently licensed to 14 countries and organizations thus far. In October 1997, Banksys agreed to cooperate technically on the Smart Access project. As a result, Belgian customers are able to pay for digital goods with their local Proton-based electronic purse in their own currency, independent of their location or the nationality of the Internet merchant. Other e-purse schemes such as the Dutch Chipper system developed by PTT Telecom and Postbank as well as the UK-based Mondex scheme have been integrated into the Smart Access service model.

One of the first Smart Access trials was launched in March 1997. The pilot used GSM access to the Internet and the Mondex payment scheme. Participating in the trial were AT&T-Unisource, Apple Computer, Nokia Mobile Phones, National Westminster (NatWest) bank, Schipol Airport, Time Out Magazine and Mondex UK. In the UK, a specially designed web site for Time Out magazine sold information on restaurants and entertainment.

Employees of Apple, Mondex and AT&T-Unisource used the GSM mobile phone link to dial-up the Internet and access the Time Out site. Once there, they could use their Mondex card to pay Time Out for information they wished to see. When payment was made, Time Out sent the information seconds later. Participants were also able to reach a Smart Access banking server to load their Mondex cards over the air by accessing their bank accounts via the Internet. This model was demonstrated further at the ITU Telecom Interactive '97 exhibition in September 1997. Nokia and AT&T-Unisource showcased wireless electronic banking over the Nokia 9000 Communicator by employing many of the same features as the UK trial.

While Smart Access employs a digital cash model to allow mobile terminals to transact for low-value goods, in Finland another transaction model for mobile handsets is being tested by national operator Sonera.

The difference between the Sonera project and Smart Access is that Finnish users settle for their selections at the end of the month on their regular telephone bill instead of using cash.

Coca-Cola Drink, a Finnish company that is 25% owned by the Coca-Cola company has introduced the 'Dial-a-Coke' concept with Sonera. Instead of paying with coins, consumers dial a short code telephone number posted on the front of a vending machine and a cola drops out. The charge for the beverage appears on the next monthly statement.

'Dial-a-Coke' grew out of a project to equip vending machines with smart phones that would ring the beverage distributor as the machine ran low on canned drinks. However, the network and distribution costs for such restricted usage did not add up. Instead, Sonera and Coca-Cola Drink decided to keep the phones inside the machines but expand the possibilities.5

The first trial for mobile-phone enabled vending machines was last summer at Helsinki International Airport. However, the transitory nature of users did not allow Sonera and Coca-Cola Drink to draw conclusions and so the pilot was moved to Helsinki University of Technology. After fifteen days of testing the mobile-phone enabled machine, Coca-Cola Drink reported that 31% of the machine's beverage sales came from mobile phones, a significant amount for a new technology.

The 'Dial-a-Coke' concept is being extended to jukeboxes, car washes and is being investigated for road tolls. The service model is heavily dependent on intelligent network (IN) technology that is able to discriminate between Sonera customers and those of rival mobile operators. The network system is connected to a billing architecture that dynamically pays the various commissions to retailers, other mobile operators and Sonera for providing the service.

A major design challenge in the future will involve numbering. By definition, the amount of numbers available to a mobile operator for identifying retail products will be in short supply. As such, Sonera is working with numbering specialists to figure out the best way to get the most number of unique products identified by the fewest numbers in a customer-friendly fashion. Should this be achieved, one can expect that Sonera will continue to find ways to exploit its mobile IN capability to extend its electronic commerce offerings or provide the platform for someone else.

Granted that Smart Access and 'Dial-a-Coke' remain in the trial stages, they are being watched across Europe. The economics of telecomms competition suggest that the trend towards mobile handsets being the main method of communications access will become the norm as mobile service packages are priced the same as fixed-line access in Scandinavian markets. As more European users discard their fixed-line connections in favor of mobile service packages, electronic commerce applications or services will have to be designed or evolved to reflect that market reality.

## III. Third Generation Future?

The first third generation (3G) mobile system will be licensed in either Finland or the UK in the spring of 1999. 3G mobile systems were originally proposed as a world-wide standard, combining the services of mobile telephony, two-way radio, paging services and broadband data capability.

The specification for a true 3G system that provides circuit-switched voice and packet-switched data at broadband speeds has been the goal for the ITU's International Mobile Telecommunications (IMT) 2000 concept. The European expression of IMT-2000 is called the Universal Mobile Telecommunications System (UMTS). A North American proposal called wideband cdmaOne is also competing for the ITU's blessing.

While different in particulars of the air interface and handset chip rates, the two standards promise to deliver voice, graphics, video and other broadband information direct to the user, regardless of location, network or terminal. Data rates of 144Kbps for wide area cellular and up to 2Mbps for fixed installations are specified in both of these standards to meet the IMT-2000 spec.

Concurrent with increased coverage and capacity is an independent service creation environment. This capability is encapsulated in the Virtual Home Environment (VHE) concept. The VHE concept means the delivery of a home operator's total service guarantee—especially for a corporate intranet—to the user at all times, wherever he or she roams in whichever network, public, private, satellite or fixed. The model calls for the terminal to negotiate functionality with the visited network, possibly even downloading software so it can mimic 'home' services in an alien environment.

The VHE network model is joined by new ways of conceiving mobile handset evolution. Ideally, a UMTS terminal should connect to the home environment as soon as it is switched on and the smart card inserted. The intervening networks, signalling, connection, log-on and any other 'technology' should be invisible to the user, so that value-added services that are bundled into the user's demographic profile are easily accessible.6

There is also the desire to make terminals and their associated smart card software downloadable over the air so that their standards and operating parameters and supported services can be modified, changed and evolved over time. Software download or software radio are usually described in the context of modifying the parameters of the radio platform (i.e. the radio modem part of the terminal). An exmaple would be the downloading of an improved handover algorithm. This aspect of software download or software radio is generally invisible to the user.

However, there is completely separate aspect of software download which is more akin to a classic client-server model. This is most easily understood in the Internet context, whereby a content provider or transaction services provider will have a specific applet which is downloaded when the user accesses the services. Examples are audio, video or graphics codecs. The two aspects of software radio will be accessed by different players in the 3G domain. The first aspect will remain the area of network providers trying to boost performance while the second will be operated by service or content providers.7

The VHE network model and software radio are considered to be where wireless e-commerce applications or services will reside. One suspects that in this environment of negotiated functionality, service providers will be qualitatively different from the simple air time resellers that dominate current second generation networks. They would probably resemble service brokers who negotiate package deals with network operators and service providers so that they can offer the user a comprehensive and comprehensible service portfolio at a single billing point.

The extent that this will include transaction capability is not known. However, it is obvious that many of the functions that are currently being trialed in the Smart Access model (for example, the catalogue server) will need to be scaled to reach an increasingly nomadic market that wishes to transact. Moreover, multifunction smart cards that contain medical or insurance data along with digital cash or credit cards can be expected to take advantage of the larger capacity of 3G systems.

NTT in Japan will launch a commercial 3G network in the year 2001 with an eye for global standardization by 2002. The NTT network is being built in cooperation with Nokia and Ericsson according to the European 3G standard. It would follow therefore, that electronic commerce services or applications providers should be aiming to provide 3G specific services before 2005. The degree that these services will be classical shopping or info-tainment or multimedia purchasing or gaming is not clear at this point.

## IV. Conclusion

In telecomms markets where prices, service quality and coverage have been geared towards consumer tastes, mobile operators have been swamped with subscribers—something that cannot be said for many other 'Information Society' applications. Additionally, mobile communications are more or less culturally neutral and cut across many more demographic groups than typical Internet applications.

Those cultural factors are joined by the fact that handsets are massively increasing their processor power over the short term. The historic bottleneck in wireless has been the air interface because radio spectrum will forever be a scarce resource. That said, advances in technology assure that mobile data transmission capability will soon outstrip that of the twisted-wire copper network. Even second generation GSM will expand its capacity from 14.4Kbps today to nearly 115Kbps by the end of 1999.

The growing convergence of fixed and mobile networks, coupled with a near universal admission by operators that IP-defined data will overtake voice as the primary communications traffic, means that the basic economies of scope for offering e-commerce services over wireless media cannot help but increase. Standardization within the IT and telecomms industries suggest that future service models will be device independent and will, in effect, be 'follow-me' service portfolios that track a user as they move across network, service and national boundaries. Terrestrial, radio and satellite systems will be compelled to cooperate to a greater or lesser degree based on a user's particular demographic profile and their willingness to pay.

That willingness to pay, in turn, is largely influenced by the service bundling and ubiquity that a service provider or broker can guarantee for the user. Given that meeting financial obligations is central to living in a market economy, it can be expected that transaction ability should follow a user as surely as dial tone or web tone.

That said, mobile electronic commerce per se remains—like most e-commerce applications—in the trial stage. There is a still a perceptual shift required for people to start depending more on their mobile terminal in lieu of physical cash or coins. 'Dial-a-Coke' is appealing but in the end somewhat limited. Smart Access is a larger step in the right direction, not because it is necessarily the most efficient system, but because it embraces the most diverse range of electronic purse schemes.

The end result for future research will be to define whether or not there are specific electronic commerce applications that lend themselves to wireless applications. The mobile terminal is the piece of network intelligence that enjoys the most favorable economics for becoming a mass-market access paradigm.

Therefore, it follows that electronic commerce systems designers would be well advised to look beyond the business models of Internet commerce delivery via a desktop PC lest they find themselves on the wrong side of history.

## Notes

[1] "Wireless Europe: Every Man and His Dog", European Equity Research Wireless Services, Salomon Brothers, March 1997, p2.

[2] Salomon, p2.

[3] Salomon

[4] Philip Barton, Zeneca Pharmaceuticals, President of the European Virtual Private Network User's Association (EVUA) , interview with author, 10 March 1997.

[5] Elina Lahtinen, Mobile Media Director, Sonera, interview with author 6 March 1998.

[6] Andrew W.D. Watson, *UMTS- Technology Vision* Motorola GSM Products Division, Swindon, UK, Chairman UMTS Forum Technology Aspects Group, presented at "UMTS-The Next Generation of Mobile", IBC UK conferences, 27 October 1997, London.

[7] Watson

# Electronic Commerce in Denmark: The Spread of EDI in Business-to-Business Transactions[1]

Niels Christian Juul, Kim Viborg Andersen, and Niels Bjørn-Andersen

*Center for Electronic Commerce, Department of Informatics, Copenhagen Business School*[2]

## Abstract

The use of EDI in business transactions is considered a prerequisite in turning both private and public business towards electronic commerce in the next century. The national strategy in Denmark has been to promote the usage of EDIFACT-based EDI in both the public and private sector. In our study of EDI-usage in Denmark, we have found this strategy only partially fruitful, so far.

We are monitoring the state of EDI usage in Denmark based on VANS, direct proprietary connections, and TCP/IP-based networks to evaluate the governmental actions within the area of EDI. Our study suggests that the governmental initiatives have been successful within the areas vital to the public services and instruments, but lagging in supporting the private sector's need for EDI and lagging in the public sector's own management.

Although Internet based EDI is growing, the business-to-business traffic through VANS, primarily using EDIFACT, hasn't been hampered. Throughout the 1995-97 period VANS traffic shows an annual increase of 33% in number of bytes and 46% in number of messages.

## 1. Introduction

Electronic commerce in both the public and the private sector involves a broad area of administrative processes and documents. A conventional dilemma for the government is whether to intervene in the diffusion and character of EDI or whether to rely on market forces to determine the manifestation of electronic commerce. In most modern economies, this ideological twist is confronted with the reality of governments' large share of the gross domestic product, which, as a consequence, both directly and indirectly affects the economy through purchasing volume, fiscal authority, etc.

This paper is concerned with why and how the Danish government attempts to stimulate the diffusion of EDI (Ministry of Research and Information Technology, 1996), thereby facilitating electronic commerce at large. Our group has initiated a continued monitoring of EDI-usage in Denmark. The initial results from this monitoring describe the developing usage of EDI from 1995 to 1998 within and between both the private and the public sector (Andersen and others, 1998b).

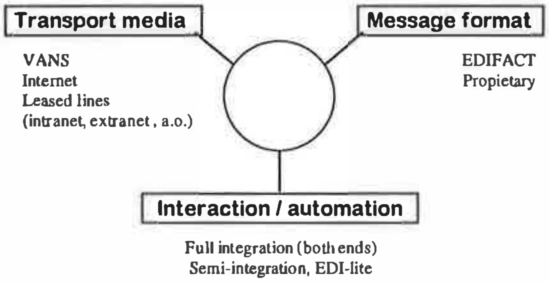
It has been suggested that there has been an uptake of Internet based business-to-business traffic for the period 1995-1997 and that the VANS operator has been a victim and unaware of this development. EDIFACT should be in the defensive and decreasing. It has also been questioned whether the public sector has been a proactive player in the diffusion of EDI. Finally, small and medium-sized enterprises should be the victims of the EDI use. We have, however, not been able to verify any of these hypothesis.

To distinguish between standards for EDI messages, the means and ownership of transport channels, the technology used in transport channels, and the integration of EDI within the IT systems of the businesses involved, a typology for EDI is presented in Section 2. The governmental instruments are briefly discussed in Section 3, whereas Section 4 presents the Danish background and the national EDI strategy. We evaluate this strategy in Section 8 based upon the survey data presented in Section 5 and examples of EDI implementation in Sections 6 and 7.

[1] This study was funded by the Danish EDI Council and the Danish National Center for IT Research.

[2] Authors address: Copenhagen Business School, Howitzvej 60, DK-2000 Frederiksberg, Denmark.
URL: `http://www.inf.cbs.dk` and `http://www.cbs.dk/cec`
Phone: +45 3815-2400 Fax: +45 3815-2401.
E-mail: `[ ncjuul | andersen | nba ]@cbs.dk`

**Figure 1. EDI Typology**



## 2. EDI, EDIFACT, and the Internet

In 1996, most of the attention was focused on application of the EDIFACT framework standard in business-to-business commerce. It opened up the possibility of solving a number of the problems that may occur in connection with sending and receiving orders and invoices and the logistics of just-in-time deliveries. UN/EDIFACT is a UN accepted global standard that makes it possible to reduce transaction costs and ensure a quick/safe delivery of data. Among the challenges in 1996 were preparing business specific manuals in application of EDIFACT, solving discrepancies in relation to the American ANSI-standard and adjusting privately agreed formats.

Manuals in the application of the EDIFACT-standard have been prepared through EANCOM (EAN Denmark). In addition to the manuals EAN Denmark also assigns location numbers to identification of a company (or parts of it) in connection with EDI transactions. If you order goods it is of course important that both the product code and order are accurate.

Since 1996 the TCP/IP technology has become so disseminated that a range of new companies and public authorities have decided to apply data entry forms, etc. to transfer orders of goods over the Internet. Transfer of data in the electronic form can also be done via proprietary formats, such as the SWIFT system used among banks.

Other central technology applications in this period comprise datawarehousing and datamining[3] in addition to developments in financial and production control systems, all of which have been important to the EDI development. Within the public sector it is especially the common filing of data from companies into a "datawarehouse" and the application of Basis Procurement that are significant in terms of EDI application. Within the private sector it is the integration modules to management accounting systems, like Concorde, Navision and SAP/R3, that enable full integration with EDIFACT. Thus, there is a range of different interests in terms of EDI application connected to the type of transport, integration and standards for the message itself, as illustrated by Figure 1.

### 2.1 *Media and Formats*

One key to the understanding of EDI implementation is to distinguish between two levels of EDI traffic:

- **The transport form**: E.g. disc, CD-ROM or tape via physical postal service, VANS operators, Internet, own network based on closed (proprietary) standards, or own (closed) network based on open standards, e.g. TCP/IP.
- **The message format**: EDIFACT-based or own (proprietary) formats.

### 2.2 *EDI Integration in Business Systems*

On this basis and in accordance with Figure 1 we have also looked at the degree of integration between data interchange and data processing in both ends of the communication. The analysis also describes the depth of the EDI application with a view to automation. Where EDI has not yet been fully integrated in both ends, we normally find systems that:

- in the initiating company's end are integrated with the company's own systems via either *proprietary standards* (P) or *EDIFACT* (E), and
- in the ends of the cooperating companies are decoupled the local systems, normally by applying *form-based EDI*, where the cooperators e.g. use a Web-browser to complete/update a form and return it electronically to the company's systems.

To the extent that a company stores data for its cooperators, e.g. by means of databases, the solu-

[3] Datawarehousing applies common storing of data, which can be retrieved by means of datamining. Oracle, e.g. provides such solutions.

tion is to the advantage of the cooperators who e.g. are saved from reentering former entered (and stored) data. The solution can also be established on the basis of business. Companies without their own EDI modules which need to exchange documents horizontally or vertically with large companies that integrate EDI fully can establish a common database between a group of member companies.

### *2.3 EDI Types*

A complete classification of the EDI application today will include a large number of categories, which we have gathered into three main categories:

1. **Proprietary standards** that include all EDI application where the messages are never based on EDIFACT, except category 3.
2. **EDIFACT-based standards** that include all EDI application where the message on its way is based on EDIFACT, including form-based EDI that is converted to EDIFACT "on the reverse side".
3. **Browser-EDI** that describes the form-based EDI which is not EDIFACT-based.

As our survey has been keen on EDIFACT, the obvious fourth category, Browser-EDIFACT, has been included in the second category when reporting numbers.

## 3. Governmental Actions

To comprehend the structural features, the initiatives at the local level of government, and the organizational initiatives, we propose to distinguish between actions by the government that are either direct or indirect in scope. *Direct actions* are targeting EDI usage by subsidizing the direct investment for ongoing EDI use, for example, by defining protocols for EDI use by customs and port authorities. *Indirect actions* capture initiatives which aim at increasing EDI "awareness" or acceptance, accomplished through public procurement as well as participation in EDI councils, international committees and special interest groups(Andersen, 1997).

We view both direct and indirect actions as having four modes (Damsgaard and Lyytinen, 1997; Andersen and others, 1998c). Whereas, the first three modes encompass the traditional government actions, organizational management is equally important:

1. pedagogical,
2. economic,
3. normative initiatives for the use of EDI in general, and
4. organizational management for the public sectors' own organizations

In many countries such as Denmark, the public sector may be a more avid user of EDI than the private sector. Consequently, government actions may affect both the demand-pull as well as the supply-push for EDI in the private sector. Accordingly, when we address the question "how does government intervention affect the EDI-diffusion process", both the public sector's diffusion and the private sector's adaptation are of equal interest(Southern, 1997; Graham and Lobet-Maris, 1994).

A plethora of instruments reflects how complicated it is to stimulate the diffusion of EDI and to estimate how government intervention affects the process. For example, the TradeNet in Singapore did not achieve success status merely by bottom-line analysis and top-down steering. On the contrast, this example showed the need to *stimulate and evaluate EDI diffusion in its organizational context*, public or private regardless. Also, *local government* and *quasi-governmental organizations* might be just as successful, or even more so, in initiating low-cost EDI solutions compared with central government.

We believe this is a very important observation given that governments have multiple forms, as well as the fact that the distinction between private and public is no longer as clear as it was decades ago. Such political and commercial changes in the environment pose challenges for the successful diffusion of EDI. However, these notions are all too often ignored (Saxena and Wagenaar, 1997; Scala and MacGrath, Jr., 1993).

Accordingly, governments should not see it as their primary role to *pursue top-down steering or legislation* of the EDI diffusion process. Using such crude strategy might stifle innovation, discourage competition, and eventually leave the national economy worse off. Instead, central government should move onward on a large variety of fronts, including fostering conditions that tear down obstacles for effective EDI use. More specific examples could be encouraging legislation for digital signatures, investing and building an information technology infrastructure, as well as organizing a watchdog against monopoly practices.

## 4. The Case of Denmark

Based on the many instruments available, a national strategy on EDI was proposed by the Danish government in 1996 (Ministry of Research and Information Technology, 1996).

### *4.1 The Importance of the Public Sector in Denmark*

The public sector in Denmark redistributes about 70% of the GDP (at factor costs), covering a variety of expenditures (unemployment insurance, housing subsidies, health care, elder care) and income sources (income taxation, corporate tax, consumer tax, import tax, etc.). In addition, about 50% of the total public expenditures are allocated through the local government (Jørgensen and Pedersen, 1994). It is estimated that the total public sector procurement is about 90 billion Danish kroner (US$13 billion).

The transfer of income is a vital part of the Danish society and is the major reason for the relatively early introduction of electronic payment transfer from the government to unemployed persons. The generous welfare system created a need for an instant check on social security numbers (CPR number), household income, documentation from doctors on medical need, etc.

The three critical characteristics of the Danish public sector—a high degree of income redistribution, highly decentralized services, and a large collection of data from companies and citizens - have created a push for the government to informate itself for technical, economic and legitimization purposes. As such, EDI has been rendered a high priority area by the central and local authorities.

Local authorities in charge of the health, education, and transportation sectors view electronic commerce as vital in all areas of procurement, as well as a means of retrieving information from citizens and commercial clients. Central government and semi-governmental units, such as the postal service and train service, perceive electronic commerce as a means to achieve strategic advances, cost and time reductions, in addition to improved overall communication. As a consequence, the Danish public sector has become the primary locomotive for the diffusion of electronic commerce relative to the private sector. In the next section we will present the overall EC strategy formulated by the Danish national council on EDI.

**Table 1. Government expenditures: Consumption, transfer, investment, other expenditures (1996)**

| | Billion Danish Kroner | Percent of GDP |
|---|---|---|
| Consumption (salary, goods and services, etc.) | 255,3 | 25,2 |
| Transfer of income | 342,6 | 33,8 |
| Investments | 22,3 | 2,2 |
| Other expenditures | 6,3 | 0,6 |
| Total Government current expenditure | 626,5 | 61,8 |

*Source*: (Statistics Denmark, 1998)

### 4.2 The National EDI Strategy

The EDI Council is the administrative unit responsible for the electronic commerce strategy. The Ministry of Research and Information Technology and the Ministry of Business and Industry grant about DKK 18 million over a 3-year period to finance the increased activities of the Danish EDI Council. Moreover, these ministries provide a pool of DKK 6.6 million over a 3-year period to subsidize standardization work, in particular, across industrial boundaries. These subsidies are distributed under the auspices of the Danish Agency for Development of Trade and Industry, as recommended by the Danish EDI Council.

The national electronic commerce strategy was formulated in 1996 in a joint effort between representatives for the public and private sector:

- Ministry of Research and Information Technology
- Ministry of Business and Industry
- Confederation of Danish Industries
- Danish Commerce and Service
- The Danish Chamber of Commerce
- Danish Bankers' Association
- The Danish Insurance Association
- Agricultural Council of Denmark
- Danish Federation of Small and Medium-Sized Enterprises
- The Danish Shipowners' Association
- The Association of Danish Mortgage Banks
- Danish Contractors' Association

The strategy has seven key areas:

1. Establishment of EDI standards in all sectors
2. EDI to be used for public procurement contracts under EU tendering by 1998
3. All public-sector financial systems should handle all commercial documents in EDI by the end of 1998
4. EDIFACT-based interchange of administrative information with the public sector in fax, statistics, etc.
5. Development of EDI software to facilitate the above
6. Legislation on digital signatures and electronic documents by early 1998
7. Danish EDI Council to act as initiator and coordinator

The strategy is joined and supported by all major players in both the public and private sector:

- Ministry of Finance's Agency for Management and Administration of Financial Affairs
- Told·Skat (the Danish Inland Revenue)
- Danmarks Statistik (Statistics Denmark)
- The Armed Forces
- Danish Palaces and Properties Agency
- DSB (Danish State Railways)
- National Association of Local Authorities In Denmark
- Association of County Councils in Denmark
- Copenhagen Hospital Cooperation
- The City of Copenhagen
- IT Trade Association
- Danish Data Association
- Danish EDI Council
- EAN-Denmark
- National Procurement Limited Denmark
- Association of Purchasers in State, Councils and Municipalities
- EDI-Building
- EDI-Transport Denmark
- Danish Pharmaceutical Association

Furthermore, the Headquarters Chief of Defense Denmark, the Danish State Railways, the Copenhagen Hospital Cooperation, the Danish Palaces and Properties Agency, as well as the Ciy of Copenhagen have used the adopted strategy by including their suppliers' ability to partake in EDI as an integral condition for inviting tenders as of 1998. Finally, the Danish government has made it official policy that all public procurement in year 2000 should be done using EDI.

## 5. Status of EDI in Denmark

We have interviewed and collected data on the use of EDI in various organizations directly involved in electronic commerce in the public and private sectors throughout the Spring of 1998. Furthermore, we have compiled data from the companies reporting data to the public sector and collected data from the VANS operators for the period 1995-1997. The data was collected between December 1997 and May 1998.

Additional data from surveys done by the Ministry of Research and Information Technology has been incorporated in this paper as well. The survey on IT in the private sector (Ministry of Research and Information Technology, 1997a) was based upon 2,001 questionnaires distributed to Danish companies. 387 were returned. Companies with 50+ employees have an over representation in the number of returned questionnaires. The survey of IT in the public sector covers the local authorities (Ministry of Research and Information Technology, 1997b) with responses from 205 of the 275 Danish municipalities and 12 of the 14 Danish counties.

### 5.1 *Basic EDI-usage*

The latest official figures from the Ministry of Research and IT are compiled in Table 2 - Table 4. These figures give an indication of how far the strategies of the organizations have been implemented.

We see that within the local government, only 8% of the municipalities used EDI, whereas 75% of the counties and 15% of the private enterprises used electronic commerce within procurement. Thus, while EDI adoption has been quite successful at the county level, substantial opportunities exist for increasing electronic transactions at the lower governmental levels.

When considering the variety of tasks in which EDI is used, approximately 50% of the counties used it to obtain information about products and services, whereas roughly 42% used it for payment. In contrast, only 5% of the municipalities used it for payment purposes. Thus counties appear to have a significant lead in utilizing electronic commerce. This is most likely attributable to the business-to-business EDI predominant in the health sector, which is primarily administrated by the counties in Denmark.

We further note that 60% of the ministries and 36% of agencies and directorates have a home page. 90% of the ministries and 53% of agencies and directorates may be contacted via an official E-mail address. 36% of the counties have a home page and 64% have an E-mail address. 16% of the municipalities have a home page and 30% have an E-mail address (Ministry of Research and Information Technology, 1997).

**Table 2. EDI in Denmark, 1997: The procurement function**

| | Yes | No | Planned 1997/98 | Total | (N) |
|---|---|---|---|---|---|
| Municipalities | 8% | 68% | 24% | 100% | (205) |
| Counties | 75% | 17% | 8% | 100% | (12) |
| Enterprises | 15% | 77% | 7% | 98% | (387) |

**Table 3. Electronic commerce: The content of EDI (percent)**

| | Information on products/services | Prices and discounts | Delivery conditions | Ordering of products/services | Payment for products/services |
|---|---|---|---|---|---|
| Municipalities | 6 | 4 | 3 | 5 | 6 |
| Counties | 50 | 34 | 25 | 34 | 42 |
| Enterprises | 15 | 7 | 5 | 6 | 2 |

**Table 4. Informatization in the Danish Public Sector (percent)**

| Level of government/ year | | No use of electronic records, management or mail systems | Electronic mail used for internal or external use | Electronic recording or electronic case administration systems introduced | Electronic handling of incoming mail and electronic case administration |
|---|---|---|---|---|---|
| Central | 1993 | 31 | 49 | 20 | 0 |
| | 1995a | 15 | 34 | 49 | 2 |
| | 1996a | 4 | 14 | 46 | 35 |
| Local | 1995b | 11 | 45 | 38 | 6 |
| | 1996c | 2 | 19 | 28 | 51 |
| Counties | 1995b | 7 | 36 | 57 | 0 |
| | 1996c | 0 | 21 | 14 | 64 |

Note. a= spring, b= primo, c= ultimo

**Table 5. EDIFACT traffic by VANS in number of bytes and messages, 1995-1997.**

| Year | Bytes | | Messages | |
|---|---|---|---|---|
| | Number | Growth per year (%) | Number | Growth per year (%) |
| 1997 | 57.110.724 | | 26.593.813 | |
| | | 33% | | 45% |
| 1996 | 43.007.936 | | 18.401.306 | |
| | | 33% | | 46% |
| 1995 | 32.375.495 | | 12.630.458 | |

Accessing the overall organizational transformation via IT, most of the applications are localized or integrated internally. However, there has been a high degree of informatization during the period 1993-1996. In 1993, for example, 31% of the central government units had no use of electronic records, management systems or mail systems, while in 1996, only 4 percent have no use of these applications. Similarly, in 1995, 6 %of the municipalities used electronic incoming mail and electronic case administraion, while in 1996, 51% of the municipalities used these applications.

### *5.2 VANS and EDIFACT*

Our survey (Andersen and others, 1998b) shows a continuous increase in number of EDI-messages transferred through VANS-operators and a sharp increase in number of companies capable of sending and receiving EDIFACT.

**Figure 2. EDIFACT via VANS 1995-1997, in number of messages per month**

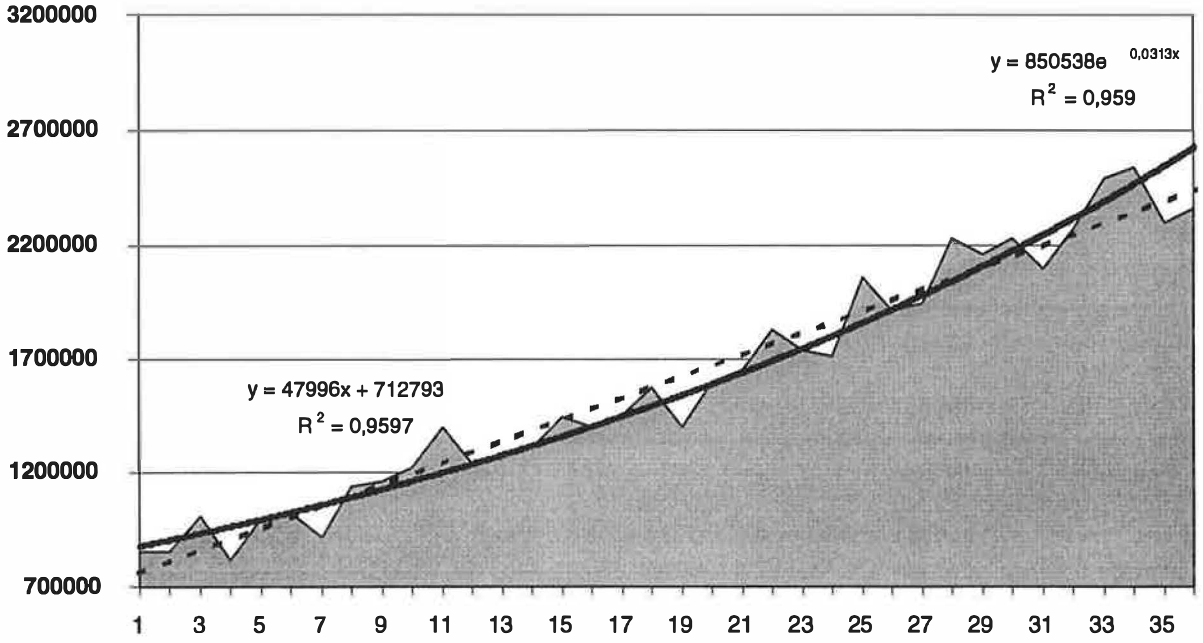


**Table 6. Allocated EAN B- og C-location numbers to enterprises, 1990-1997**

| Year | Number of newly allocated numbers | | Accumulated amount of B and C numbers | |
|---|---|---|---|---|
| | B-number | C-number | Total | Index (1990=100) |
| 1990 | 0 | 16 | 16 | 100 |
| 1991 | 2 | 45 | 63 | 394 |
| 1992 | 25 | 66 | 154 | 963 |
| 1993 | 55 | 53 | 262 | 1.638 |
| 1994 | 59 | 50 | 371 | 2.319 |
| 1995 | 45 | 45 | 461 | 2.881 |
| 1996 | 77 | 42 | 580 | 3.625 |
| 1997 | 146 | 34 | 760 | 4.750 |
| Total | 409 | 351 | n.a. | n.a. |

Note. *B-numbers are EAN-location numbers sold without the associated manual. C-numbers are EAN-location numbers including manual.*

As shown in Table 5, more EDIFACT messages have been sent via VANS measured both in terms of number and size. From 1995-1997 the EDIFACT application has increased on average by approx. 45% annually in terms of number of messages and by approx. 33% in terms of size of messages. If we compare the number of bytes via VANS in EDIFACT-format in December 1997 with the number in January 1995, the increase amounts to approx. 82% over the three years. Measured in terms of messages, the study shows that *176%* more messages were sent in December 1997 compared with January 1995 (Figure 2).

Table 6 shows the sharp increase in the number of companies capable of sending and receiving EDIFACT messages. The analysis of the number of users of the EDIFACT standard in Denmark shows a particular increase in the number of small companies among the new users of EDIFACT. From 1995 to 1996 the total number of companies that could send/receive EDIFACT messages increased by 26 percent compared to 31 percent from 1996 to 1997. Among the small companies the increase amounted to 90 percent from 1996 to 1997.

## *5.3 Internet Based EDI*

Rather than using VANS operators, one might use the Internet for EDI. Approximately 50% of Danish enterprises with more than 5 employees have access to the Internet. Almost 80% of the enterprises communicate electronically from the enterprise via call connections or leased lines (Ministry of Research and Information Technology, 1997).

At present, it is very limited how much companies apply browser-EDI for business-to-business trade. Only five per cent of the companies have used the Internet for electronic commerce, whereas approx. 35% of the companies use EDI via VANS or proprietary (closed) circuits. As yet, the companies are not convinced that delivery is ensured, nor secured, on the Internet.

It is predicted that browser-EDI, which can be called a relatively inexpensive integration of the extreme ends of the value chains, will continue to make headway. The end of the communication where the EDI, possibly the EDIFACT message, is integrated into the company's other business processes will benefit substantially from this. Therefore, the solution must be seen as a first step towards full integration in the "browser end" in order that data also can be interchanged electronically with the other business processes in this end.

The companies will increasingly apply browser-EDI so they themselves can offer "VANS-services" and services/sales via Web pages on the Internet. According to our estimates this will not reduce the number of companies that apply EAN-numbers or the number and the size of EDIFACT-transmissions.

## *5.4 Distribution of EDI Types by Sectors*

Based on the interviews in industry associations and various other available data we have made a quantitative estimate of the status of the EDI application as shown in Table 7 and Table 8. In the headings of the tables the electronic interchange of data has been divided into three format types: proprietary EDI, EDIFACT and browser-EDI. The column to the extreme right shows the growth potential for electronic communication, which is defined as the remaining quantity of documents that have not yet been converted to EDI. The sum of the four columns equals 100%.

The tables' extreme left column shows industry areas. The cells of the tables show our estimate of the present coverage in percentages at the beginning of 1998 for the three categories in addition to our estimate of the annual increase in the period 1998-2000. In the insurance area less than 10% of communication is made by means of proprietary standards, more than 50% of communication takes place via EDIFACT, while there is less than 1% application of browser-EDI internally in the industry. In the next 2 – 3 years the industry expects EDIFACT- application to increase by 10-20% and browser-EDI application by 20-30%.

**Table 7. Dissemination of EDI in the financial sector, production and commerce, and the transport sector**

| Sector | | EDI type | | Potential |
|---|---|---|---|---|
| | Proprietary EDI | EDIFACT | Browser-EDI | (not realized) |
| Insurance | <10% coverage | >50% coverage<br>(10-20% growth) | < 1% coverage<br>(20-30% growth) | 30-40% |
| Banks | 70-80% coverage | <2% coverage<br>(10-20% growth) | <0,1% coverage<br>(5-10% growth) | 20-30% |
| Mortgage credit | 70-80% coverage | EDIFACT-based Intranet from 1999 | < 1% coverage | 20-30% |
| Industry | n. a. | 15% coverage<br>(15% growth) | < 1% coverage | 80-85% |
| Wholesale | n. a. | <5% coverage<br>(10-50% growth) | < 1% coverage | >95% |
| Crafts and small industries (SMEs) | n. a. | <5% coverage<br>(10-50% growth) | < 1% coverage | >95% |
| Construction | n. a. | <5% coverage<br>(10-50% growth) | < 1% coverage | >95% |
| Retail commerce | n. a. | 20-30% coverage<br>(20-0% growth) | < 1% coverage | 70-80% |
| Agriculture | 30% coverage | 30% coverage<br>(10-20% growth) | < 10% coverage | 30-40% |
| Heavy freight | >30% coverage | >30% coverage | >10% coverage<br>(20-30% growth) | 30% |
| Postal services | >20% coverage | 20-30% coverage* | <10% coverage<br>(20-30% growth) | 40% |
| Shipping | >30% coverage | >30% coverage | >15% coverage<br>(20-30% growth) | 20-30% |

*Notes:* The growth in brackets denotes the expected annual growth in messages measured in percentage of the previous year for the period 1998 - ultimo 2000.

n. a. means "not available".

* domestic messages; but >90% coverage of non-domestic messages.

**Table 8. Dissemination of EDI in the public sector**

| Authorities | EDI type | | | Potential |
|---|---|---|---|---|
| | Proprietary EDI | EDIFACT | Browser-EDI | (not realized) |
| Central level | <10% coverage | 10-20% coverage | 10-20% coverage<br>(10-40% growth) | <50% |
| Local levels (counties, municipalities) | <10% coverage | <20% coverage | <10% coverage<br>(10-40% growth) | <60% |
| Public Health | <10% coverage | 20-30% coverage | <1% coverage<br>(5-10% growth) | 50-60% |

### *5.5 EDI Usage by Type*

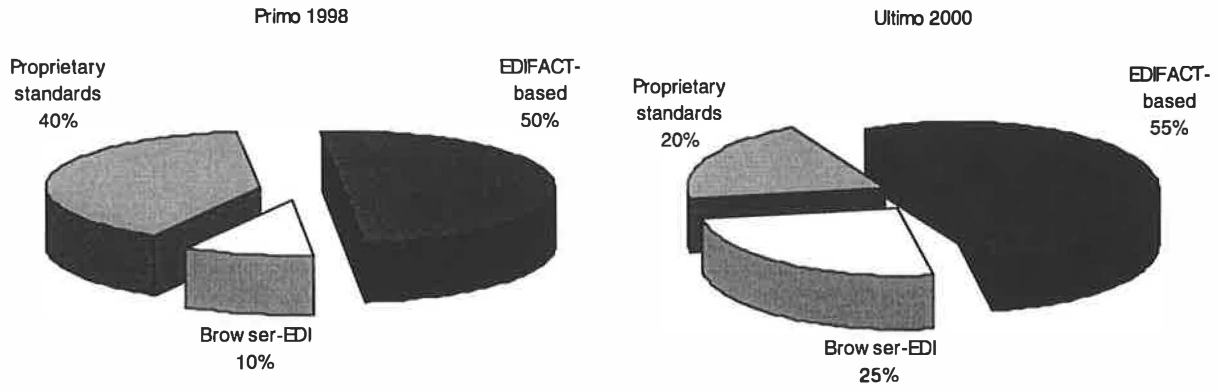
It has not been possible to compile enough data to give a precise summary of the current distribution of EDI usage among our three EDI classes. A prediction of the future distribution is, of course, even harder to make. Based on our current information, a yearly growth in number of transactions between 30% and 50% is not unlikely.

Compared to the total amount of EDI-usage, Browser-EDI is especially likely to raise its share (from 10% to 25%), while the EDIFACT-based EDI is only expected to raise from 50% to 55% in the 1998-2000 period. Consequently the share of proprietary standards is expected to deminish from 40% to 20%. Figure 3 summarizes the distribution of EDI usage among our 3 EDI types by early 1998 and by the end of year 2000.

## 6. Business Associations Taking Part in the EDI Strategy

In general, the business associations assess that it is first and foremost order and invoice messages that are exchanged in EDIFACT-format. At present, small companies have great expectations to current development projects that focus on EDI solutions by means of the Internet technology, the so-called form-based EDI solutions. Several VANS-operators are involved in the work. It is characteristic for companies that have implemented EDI that their interchange involves relatively few partners and a relatively small number of messages.

**Figure 3. EDI-usage by proprietary standards, EDIFACT, and Browser-EDI in 1998 and 2000**



### 6.1 Insurance and Banking

The banking sector has a long tradition for EDI application, but less than 2% of all EDI transactions are EDIFACT-based. They are especially used in transactions between banks and large customers. EDIFACT-standards have been prepared for all important financial transactions between banks and their customers.

The mortgage credit sector is facing a big commitment when they launch their mortgage credit network in 1999, an EDIFACT-based Intranet solution[4] that will combine all parties within mortgage banking: the seven mortgage credit institutions, legal advisers and intermediary lenders. However, the customer will not be linked to the system, as it will be up to the individual mortgage institutions and intermediary lenders to make their own image in relation to customers via open Internet solutions.

### 6.2 Industry and Commerce

Within the Danish Chamber of Commerce, which numbers approximately 2,000 member companies, less than a handful of companies make transactions via VANS at present. The only companies that apply EDI are found within the food industry and the pharmaceutical area with MedCom, which corresponds to less than 0.1% of the Chamber of Commerce members.

Contrary to this, all large and medium-sized insurance companies participate in the EDI exchange of the two EDIFACT-documents that are in production within the insurance sector. The interchange comprises approximately 80% of all transactions. The remaining 20% derive from small insurance companies that until now have found an EDIFACT-solution too expensive on account of the price of EDIFACT-converters. The insurance industry plans a data-entry based solution by means of the Internet technology, which can convert messages to/from EDIFACT-format. Apart from the EDIFACT exchange, the insurance industry – like the banking sector – has applied EDI in many years, both internally in the industry and in exchange with other sectors, such as the banking community.

Within the agricultural sector there is a substantial EDI application in relation to the veterinary authorities and the consultative institutions that need accounts data. This development is especially due to the large efforts made by LEC. At present, LEC is working on developing EDI application within e.g. the export restitution area. The Agricultural Council in Denmark has also indicated that the industry association cannot push their members too much to make them use EDI.

### 6.3 Transportation

In a similar way, the Shipowners' Association and the Danish Provisions Suppliers' Association (DLF) point out that they do not play a very active role; but wait for the largest players to take action. There is no need for special initiatives in this area.

DLF is involved in the technical part of the EDI application due to their members' interests in EDI and their membership of EAN Denmark. DLF represents the producer link (approximately 150 members), and consequently the association is very involved with VANS-operators. As an industry association for such companies as MD Foods, Carlsberg and Danisco DLF plays a central role in relation to EDI application in the supply chain and the business relations. The problems that are taken up involve primarily the physical transport of goods. There is a need for coordination within logistics, distribution, marketing and sales.

The data communication between DLF's members and the provisions trade has grown substantially. Due to the many products that are transported daily, there is a large need for data. It is the producers in cooperation with the provisions trade that have made the EDI trade standards. DLF has appointed a EDI committee that is taking care of members' interests in relation to pace and direction of EDI application. Since 1995, the number of members that apply EDIFACT has increased by approximately 20%. Approximately 75% of the users apply invoice and order documents. There are 15% more trading partners who apply EDI in 1997 compared with 1995.

## 7. EDI Application in the Danish Public Sector

In the previous sections we have reviewed both the context and the status of informatization in Denmark through 1996/1997. We will therefore proceed to pre-

[4] EDIFACT-messages are transported by means of TCP/IP in a new closed Intra-/extra-net.

sent some of the prime examples of EDI use in the Danish public sector in order to illustrate how the public sector is the main driver in EDI diffusion.

## *7.1 The Danish National Board of Industrial Injuries*

The Danish National Board of Industrial Injuries is a financially autonomous agency under the Ministry of Social Affairs handling individual work related injuries. The work process involves retrieval of data, checking of legal data/ reference materials, and cooperation with other colleagues in the office. Most of the external contact is done through traditional mail (incoming mail is scanned) and phone. Within the organization communication is either face-to-face meetings, phone meetings, memos, and exchange of electronic documents. In 1995, the Board received 327,000 letters, while they sent off 375,000 letters. They received 46,000 notifications of injuries and made 90,000 decisions.

In 1997, the insurance companies and the Board started using EDI. The insurance companies are required by law to use the board in cases of worker compensation related to injuries. The Board needs to check insurance numbers, the insurance companies need to check social security numbers. By using EDI, the two partners have bypassed a legal barrier, which prohibits insurance companies from obtaining direct access to centrally stored personal data.

## *7.2 Public Procurement*

The most demanding challenge for the near future is to implement EDI based electronic trade into public procurement. While the need to obtain cost reductions in administrative processes is evident, the barriers of technique, tradition and attitude are still pervasive.

National Procurement has assumed a major role in preparing the public sector in Denmark for the electronic procurement of goods and services. In 1997, National Procurement Ltd. introduced an EDIFACT based database which encompassed all goods and services in the current paper based Procurement System. In addition, an electronic public procurement system has been designed in cooperation with the central and local public network operators.

These two systems are completed with EAN location numbers for all subscribers as well as a set of EDI documents and standards which will form the backbone of a thorough public trade environment. This will be combined with an open interface to other 3rd party goods and services databases and administrative systems used in the public sector

In the course of 1998 all public authorities will be able to commence transacting via EDI. Together with National Procurement, Ltd., the Agency for Financial Management and Administrative Affairs is developing a basic procurement system for public financial management systems. The procurement system became operative for local and central government users as of February 15 1998.

## *7.3 Taxation*

The Danish Central Customs and Tax Administration (Customs*Tax) also aims to receive all documents electronically. Their EDI strategy consists of two major elements: 1) A strategy for the handling of all incoming data from companies electronically through an alliance with the Statistics Denmark and the Danish Commerce and Companies Agency, Ministry of Business and Industry; 2) enabling citizens to deliver their advance tax assessments and income tax statements via the Internet and voice-response.

The requirements for EDI applications used by Customs*Tax customers are:

1. Right amount on time.
2. Better servicing of companies, including a reduced effort on their behalf.
3. Effective use of resources.
4. Accurate and lawful administration.
5. Development oriented.
6. Political satisfaction.

The new EDI interface developed by Statistics Denmark, Danish Commerce and Companies Agency, the Ministry of Business and Industry, and the Central Customs and Tax Administration enables companies to deliver their declaration regarding VAT and excise duty electronically. In 1997, more than 20% of all Danes made use of the service to enter and transmit information for the advance tax assessment and the income tax return by telephone. Less than 1% used the Internet for this task in 1997.

**Table 9. Central Customs and Tax Administration: Documents handled manually and electronically, 1997 (1,000)**

| Document: | Manual data entry | Electronic data transfer | |
|---|---|---|---|
| | | EDI | Phone |
| Employers' income statement | 1,000 | 10,000 | 0 |
| Interest (financial institutions) | 150 | 31,000 | 0 |
| Trade union dues | 100 | 5,300 | 0 |
| Income tax statement | 780 | 30[a] | 600 |
| VAT | 1,600 | 0 | 0 |
| Intrastat | 900 | 4,000 | 0 |
| Import and export, total | 2,800 | 1,100 | 0 |

a) Via Internet

## 7.4 Health Sector

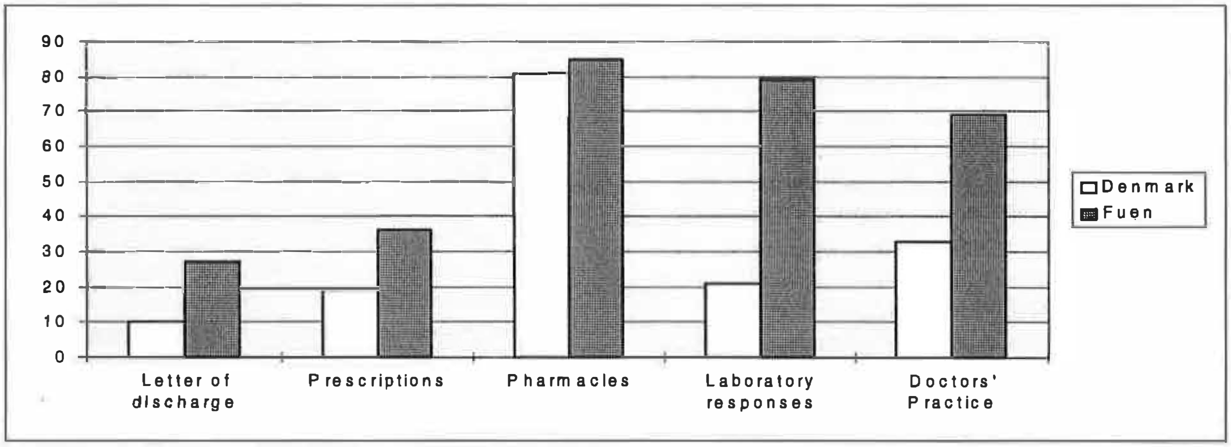
The Danish health care sector is considered to be one of the most automated in Europe. Currently, 15% of discharge letters, 7% of laboratory results, and 10% of pharmacy prescriptions are handled by EDI on average. In the Copenhagen city, EDI is also used in the communication with dentists and opticians.

Within Fuen county, the general practitioner (GP) sends a referral to the hospital department, and the hospital sends discharge letters to the GP and the municipal social administration and the health insurance using EDI (Andersen, 1998a; Fuen County, 1995). Also, when the pharmacy, hospital and GP order medicine, EDI is replacing paper based order and invoice forms. Currently, the FuenCom project is being expanded to cover the entire country (MedCom).

After establishing a computerized health data network within the organization, the county estimated that substantial revenue had been achieved (Fuen County, 1996). There is, however, major variance in how much of the communication is wired. In Figure 4, we compare the communication in Fuen and the average communication in Denmark based on the number of letters of discharge, prescriptions, pharmacies, laboratory test results, and doctor's practices.

Twenty-seven percent of the letters of discharge are computerized and are part of a network in Fuen County. Yet, at the national level, only 10 percent of the letters of discharge are part of a network. Most remarkable, however, is that almost 80 percent of the laboratories send and receive their messages through electronic communication in Fuen County, whereas only 20 percent of the communication in the labs are processed electronically at the national level. Furthermore, more than 4 out of 5 pharmacies are wired. This is true for Fuen County as well as the national average.

**Figure 4. Network Distributed Messages in Fuen and in Denmark on Average**



*Note.* Data provided by the Danish Center for Health Informatics.

## 8. Conclusion

The Danish government has launched a very ambitious plan in 1996 to expand electronic commerce in general and the procurement function in particular by introducing the technical capability by 1998 to realize electronic commerce at all levels of government by the year 2000. Our data shows that the central government is on the right track and likely to succeed within procurement. In other areas, such as the health sector and taxation, the strategy is successful as well. Not only is the diffusion rate very high, the public sector has also been innovative in co-developing the EDIFACT standards within the taxation, health, and insurance areas.

One of the issues that fascinates us is the Danish public sector's role as a major market player in buying goods and services, which makes electronic purchasing and procurement vital in fulfilling the goals of efficient cost-management. Second, we found that while the government has been using EDI instruments at the top-levels of government, the initiatives often comes from the *lower levels of government.*

The government has, by following a multi-dimensional strategy combining direct and indirect actions, stimulated the use of EDI, not only in their own organization, but also at other levels of government. In addition, they are applying normative, pedagogical and economic actions to enhance EDI in the communication with their business partners.

Although the governmental initiatives have been successful within the *areas vital to the public services and instruments*, they are lagging in support of the private sector's need for EDI as well as the public sector's own management. Examples are numerous: *on-line systems* within an organization, information systems for *joint use horizontally and vertically within the public sector*, EDI/ data network *with the suppliers* of taxes, goods, etc. *and the citizens*, *international network* in the criminal justice area, etc. These information systems present challenges for the system manager, but also, from our perspective, for the core functions within the public administration and its network.

Moreover, in the intermediate future, we see problems emerging for governments if they continue to focus upon control and power, rather than favorable relationships with trading partners. For example, building national and supranational data networks where EDI is the major vehicle will confront political challenges such as privacy issues. With the open exchange of legal and health data, total privacy can never be completely assured using EDI and IOS. However, networks must not be constructed to maintain or reinforce governmental power-balances, rather, the goal must be to demonstrate efficiencies (human labor savings, savings in time and transportation, better utilization of capital in scheduling, etc.), for the large investments in EDI.

## 9. Acknowledgment

The study of EDI in Denmark has been made at the request of the Danish EDI Council who asked us to make a survey of the EDI development in the period 1995-1997 and to give a prognosis of EDI application in the years to come.

We are greatly indebted to the VANS-operators, ant to the companies and public organizations for their readiness to cooperate and participate in the interviews and data collection.

The research assistants, Lotte Mangor, M.Sc. student in International Economics and Management, Tine Bjørn-Andersen, M.Sc. student in Economics and Business Administration and Nina Christiansen, M.Sc. student in Business Administration and Computer Science, have assisted in the data processing of the material collected. MA in translation, Birgitte Bush has been most helpful with translation.

## 10. References

Andersen, Kim Viborg. (1997). *EDI and Data Networking in the Public Sector: Governmental Action, Diffusion, and Impacts.* Amsterdam: Kluwer.

Andersen, Kim Viborg. (1998a). Health Data Networking: Capability, Interaction, Orientation, and Political Values. *Hawaii International Conference on Systems Sciences (HICCS)*

Andersen, Kim Viborg, Bjørn-Andersen, Niels, and Juul, Niels Christian. (1998b). *Elektronisk handel og dokumentudveksling: EDI-anvendelsen i den private og offentlige sektor.* Frederiksberg, Danmark: Samfundslitteratur. [Electronic Commerce and Document Interchange: EDI Applications in the Private and Public Sector] (In Danish) English summary available on the Web: `http://www.inf.cbs.dk/inf/projects/edi`

Andersen, Kim Viborg, Bjørn-Andersen, Niels, and Wareham, Jonathan. (June 1998c). Using the Public Sector as a Locomotive for Electronic Commerce: The Case of Denmark. *Proceeding of the 11th International Conference on Electronic Commerce*, Bled, Slovenia

Damsgaard, J. and Lyytinen, K. (1997). *Government Intervention in the Diffusion of EDI: Goals and Conflict.* Chapter in *EDI and Data Networking in the Public Sector: Governmental Action, Diffusion, and Impacts* Andersen, Kim Viborg, ed. Amsterdam: Kluwer,

Fuen County. (1995). *FYNCOM: Det fynske sundhedsdatanet.* Odense, Denmark: Danish Center for Health Informatics. [The health data network in Fuen] (In Danish)

Fuen County. (1996). *MEDCOM: Det danske sundhedsdatanet.* Odense, Denmark: Danish Center for Health Informatics. [MEDCOM: The Danish health data network] (In Danish)

Graham, I. and Lobet-Maris, C. (1994). *EDI impact: Social and economic impact of electronic data interchange.* TEDIS Project C9 ed. Brussels: European Commission.

Jørgensen and Pedersen. (1994). *Den offentlige sektor.* Copenhagen: Handelshøjskolens Forlag.

Ministry of Research and Information Technology. (1996). *Electronic Commerce in Denmark - a national EDI action plan.* Copenhagen, Denmark: Ministry of Research and Information Technology. Also published on the Web: `http://www.fsk.dk/fsk/publ/elcom/`

Ministry of Research and Information Technology. (1997). *Authorities Heading for a Fall: IT Policy White Paper Presented to the Folketing.* Copenhagen, Denmark: Ministry of Research and Information Technology. Also published on the Web: `http://www.fsk.dk/fsk/publ/1997/autoriteter/uk/`

Ministry of Research and Information Technology. *Danske virksomheders brug af IT 1997 (In Danish) [The Use of IT in Danish Enterprices].* 1997. Available on the Web: `http://www.fsk.dk/fsk/publ/1997/it97.html`

Ministry of Research and Information Technology. *IT i kommuner og amter 1997 (In Danish) [IT in Municipalities and Counties 1997].* 1997. Available on the Web: `http://www.fsk.dk/fsk/publ/1997/it97-kom-amt/`.

Saxena, K. B. C. and Wagenaar, R. W. (June 1997). Critical success factors of EDI technology transfer: A conceptual framework. *Proceedings of the 3rd European Conference on Information Systems (ECIS)*, Athens, Greece., 57-74.

Scala, S. and MacGrath, R., Jr. (1993). Advantages and disadvantages of electronic data interchange. *Information & Management* 2: 85-91.

Southern, A. (1997). *EDI, Information Processing and Issues of Governance: The Informational Character of UK Local Government .* Chapter in *EDI and Data Networking in the Public Sector: Governmental Action, Diffusion, and Impacts* Andersen, Kim Viborg, ed. Amsterdam: Kluwer,

Statistics Denmark. (1998). *Danmark i tal.* Copenhagen: Statistics Denmark. [Denmark by the numbers] (In Danish)

# Notes from the Second USENIX Workshop on Electronic Commerce

Michael Harkavy, Andrew Myers, J. D. Tygar, Alma Whitten and H. Chi Wong

*Carnegie Mellon University*
*Pittsburgh, PA 15213*

## Abstract

*These are notes taken from the Second USENIX Workshop on Electronic Commerce from November 1996. They record presentations and questions from this workshop.*

## Introduction

The Second USENIX Workshop on Electronic Commerce was held in Oakland, California at the Claremont Hotel, with a day of tutorials on November 18, 1996, and three days of technical program beginning on November 19.

The best paper award was won by Ross Anderson and Markus Kuhn for "Tamper Resistance — A Cautionary Note." The best student paper award was won by David Wagner for his submission with Bruce Schneier, "Analysis of the SSL 3.0 Paper." (Please check with the authors for final version of this paper.)

The proceedings of the First USENIX Workshop on Electronic Commerce (which appeared several months after the workshop was completed) included a set of notes prepared by Peter Honeyman and his students. This guide was useful to researchers, so J. D. Tygar and his students organized a similar note-taking exercise for the Second Workshop. Since the proceedings for the Second (and Third) Workshops were distributed during the workshop, the program chairs for the Second and Third Workshops agreed to have these notes appear in the proceedings of the Third Workshop.

Electronic commerce has changed a lot since November 1996. We hope that the attendees to the Third USENIX Workshop will enjoy seeing this snapshot of the research state of our field from two years ago.

## Session I: Hardware Tokens

### Tamper Resistance – A Cautionary Note

*Ross Anderson, Cambridge University; Markus Kuhn, Purdue University*

Markus Kuhn began by pointing out that, while cryptographic security usually assumes that attackers can't get at the secret keys or observe the computations, current distributed and mobile applications such as pay TV access control give attackers plenty of access to the hardware. He stated that he would discuss hardware security and tamper resistance in terms of three classes of potential attackers: clever outsiders, knowledgeable insiders, and funded organizations.

Markus then described a host of simple attacks on the physical security of smart cards. The tamper resistant coating on the Motorola smart card chip can be dissolved with $30 worth of fuming nitric acid and acetone. Access to software stored in standard microcontrollers is generally prevented by setting an irreversible security fuse bit, but a UV EPROM eraser can be used to reset the security fuse and the software can then be read. Special smart card security processors usually have a melt fuse as the security bit, but a well-equipped lab can often repair the fuse. For many microcontrollers voltage attacks can successfully reset the security bit. Other techniques for accessing the software include timing analysis, applying heat gradually to toggle EEPROM bits, and recording current leakage. It is also possible to change single instructions by signal glitches such as increasing the clock frequency; for example, a loop control variable can be changed causing additional memory content to be output.

Markus stated that all these attacks are feasible even for clever outsiders. Knowledgeable insiders and funded organizations who have resources up to $50,000 may have access to tools such as microprobing workstations and laser cutters for breaking connections and removing the passivation layer. If they have up to $1,000,000 available, they may use electron beam testing for reading bus signals, focused ion beam workstations for making new connections on the chip, and selective dry etching. If they have even more resources, they may use tools like automatic layout reconstructions to recreate circuit diagrams, electro-optic sampling and IR rear access.

Markus concluded by stating that the moral is not to blindly trust manufacturer claims about tamper resistance, avoid global secrets, reduce the importance of tamper resistance whenever possible, use fault-tolerant machine code in smart cards, implement fallback modes and insist on in-depth hostile review of designs.

## Token-Mediated Certification and Electronic Commerce

*Daniel E. Geer and Donald T. Davis, Open Market, Inc.*

Dan began by noting that public key cryptography usually expects the use of certification authorities (CAs) to remove the need for real-time authority participation; he then suggested that we consider what could be done if certificates were made really cheap and disposable. For example, a smart card in the wallet could be trusted to act as a CA and generate certificates on a per-transaction basis, not to supplant existing CAs and certificates, but as a supplement. This could allow us to get highly scalable access control and simpler payment protocols.

As one possibility, we could have a delayed purchase scenario, in which the goods are not currently available at the time of purchase. We generate a certificate for the merchant which is a public key, an authorization to deliver (download) goods, and an expiration. The raw material required for this would be a smart card with reader, a cryptographic coprocessor, a secure key store, browser support for smart cards such as a Netscape plug-in or Microsoft cryptographic API, and certificates such as X.509v3. No certificate directory or revocation lists would be required. The card owner would begin by generating two key pairs, one for the owner and one for the card, and would certify the card's key with the owner's key. The private keys could then be deposited with a key recovery center, cross-encrypted with the public keys.

Dan argued that there would be many advantages to such a system. For authorization, he stated that such certificates correspond naturally to roles, and that roles scale better than access control lists and are easier to think out properly than capabilities. He pointed out that revocation would be unnecessary because these certificates are both short-lived and not generally published. He listed advantages of this scheme for electronic commerce including: ease of set up, design and management; the ability to do delayed fulfillment such as prime-time purchases with off-time deliveries; fewer on-line parties needed for each transaction; no access control list management; new services that rely on asynchronous delivery, such as magazine subscriptions; and consumer ease and safety because delivery can take place securely without the participation of either the customer or the smart card.

Marvin Sirbu pointed out that a Kerberos ticket and session key could be used instead of a public key; Dan agreed that public key cryptography was unnecessary here and that Kerberos would work fine. Greg Rose asked why it's necessary to create a new key pair; Dan's answer was that it's a containment issue. Terry Ingoldsby asked why the bank can trust the merchant to take payment from the customer's account; Dan explained that the certificate given to the merchant is signed with the customer's private key.

## Smart Cards in Hostile Environments

*Howard Gobioff, Carnegie Mellon University; Sean Smith, IBM Research; J. D. Tygar, Carnegie Mellon University; Bennet Yee, University of California, San Diego*

Howard Gobioff described a security problem with smart cards due to the lack of direct I/O to the customer; that is, in an untrusted environment all communication between the customer and the smart card must go through an untrusted card reader. As an example of the problems this can create, Howard described a point-of-sale scenario in which the merchant's terminal reports the transaction to the smart card as $100 while displaying it to the customer as $10.

We would like to have communication between the customer and the smart card be secure (private and trusted) in both directions. Howard outlined possible additional capabilities that we could assume for the smart card, and the benefits that would result from each.

If we assume that we have a one-bit private input channel from the customer to the smart card, then we can also have private output from the smart card to the customer, by having the customer input a key through the private channel which the smart card can then use to encrypt its output. This might be used to allow the customer to check the balance in the smart card without revealing it to the merchant. Similarly, if we assume we have a one-bit private output channel from the smart card to the customer, then the card can provide the customer with a key and the customer can input encrypted values for privacy. This is similar to work by Abadi, Burrows, Kaufman and Lampson in which the card presents a random value which the customer then adjusts using arrow keys.

If we assume we have trusted input plus one bit of trusted output, then we can have general trusted output: the customer feeds the displayed value back to the card via trusted input and the card uses the one bit of trusted output to signal any discrepancy. Likewise, if we assume we have trusted output plus one bit of trusted input, the card can display and the customer can signal discrepancies, so again we have general trusted output.

Bob Gezelter pointed out the importance of having a timeout equal not-okay, since otherwise the merchant can attack by distracting the user at the right time. Simon Kenyon argued that in a closed-loop system fraud will be caught when the books are balanced; Howard responded that this is true in present systems but fraud is still a problem.

# Session II: Protocol Analysis

## Analysis of the SSL 3.0 Protocol

*David Wagner, University of California, Berkeley; Bruce Schneier, Counterpane Systems*

SSL is a protocol for practical application–layer security, mostly for web traffic. The most recent version, 3.0, is an attempt to fix problems with version 2.0 and to add support for more cryptographic algorithms.

SSL version 3.0 is, overall, an improvement. The MAC keys have been expanded to 128 bits, even in the exportable version. Separate keys are now used to perform encryption and authentication. Finally, SSL now uses HMAC as its message authentication algorithm. These improvements help to stop replay and connection truncation attacks. Some standard, simple attacks were tried, but none compromised 3.0's security.

One attack that is successful against 3.0 is a traffic analysis attack, based on the observation that the ciphertext length usually reveals the plaintext length. An eavesdropper on web traffic could record the encrypted request for a URL and a server's encrypted reply. If the attacker can get an index of all the documents on a web server, he can compare the encrypted lengths of the documents and their request strings to the lengths of the observed request and document. A match in lengths between the lengths of an encrypted document and the observed document as well as the encrypted document request and the observed request indicates a very likely match between the unencrypted forms of the documents. This attack reveals which documents a client received from a web server.

In addition, there is an attack on the handshake layer of the protocol. A field used to select between the RSA and Diffie–Hellman algorithms is not part of the signed section of a message. If an adversary were to change the value of that field, the client could become confused about the meaning of signed data, eventually compromising the session's security. If the client performs a sanity check, it could detect this attack, but the sanity check is not mandated by the protocol. The suggested fix is to expand the signature to also cover the field.

There are still many questions for future study. David is not sure whether or not all nonces are properly signed, and the protocol should be checked to make sure that it is not vulnerable to session resumption and version rollback attacks. In general, this analysis was informal, not formal, meaning that it can only illustrate flaws in the protocol, not prove that it's correct.

Martin Abadi asked what the maximum amount of an SSL transaction in which David would be willing to participate was. David answered that SSL is probably acceptable for encrypted credit card transactions. Dan Geer was curious about where the most likely points for an implementation to fail would be. David responded that sanity checks and key management were likely places. Another question was whether cataloging all the documents on a web site is really possible. David did not know how much of a typical web site is made up of dynamically generated documents, but he suspected that the amount is non–trivial.

## Fast, Automatic Checking of Security Protocols

*Darrell Kindred and Jeanette Wing, Carnegie Mellon University*

An increasing number of security protocols are becoming more important. These protocols are being developed extremely rapidly as well, and it's hard to get these protocols right. We would like to automate the protocol–checking process to save time and gain more confidence in the results. There are several approaches to verifying that a protocol is correct. Handwritten proofs are easy to get wrong. Another approach which is effective, though tedious, is to reason about a protocol in high–order logic and use theorem proving software to verify correctness. Finally, small logics have been developed for reasoning about parts of protocols, e.g. BAN. The usual answer from such logics is limited to yes, no, or cannot decide.

The focus of this work is on improving automated support for using small logics. In such logics, a protocol message is represented as a logic formula reflecting the result of the message being received, a set of assumptions, and a set of inference rules to express familiar concepts. Usually, the premises of the rules are larger than the conclusions. Thus, the process of generating conclusions will terminate eventually.

We can exhaustively produce all truths in the protocol. With these truths, we can check that properties hold and explore what effect changes in the protocol will have on the truths generated.

A logic checker was implemented which can verify that a logic satisfies certain restrictions. In the actual checker, there are three types of rules: shrinking, growing, and rewrites. The checker uses shrinking rules to generate truths, only applying other rules when no more shrinking rules can be applied. This method does not generate all truths, but typically, interesting truths are generated. We can also check whether a specific formula is derivable.

Currently, the logic checker should be useful for protocol developers; it is automatic and fast, with most runs being on the order of one to two minutes. In the future, more logics will be developed to reason about other properties such as anonymity or safety from man–in–the–middle attacks, and support for temporal logics will be added.

Nevin Heintze was curious as to whether, given a property, a system could work backwards, giving changes to a protocol which would be required for the property to hold. Darrell pointed to model checking, which gives a counter–example when a property does not hold, and said that getting the smallest set of changes to a protocol that would make a property hold would be possible.

### Verifying Cryptographic Protocols for Electronic Commerce

*Randall W. Lichota, Hughes; Grace L. Hammonds, AGCS, Inc.; Stephen H. Brakcin, Arca Systems, Inc. Presented by Jack Wool.*

The aim of this work is to develop methods for verifying protocol correctness that will aid protocol designers at the beginning of the design process, rather than later, after the protocol is already in use. The focus is on determining what each party in a transaction can be proved to believe. The tool should be usable by protocol designers, not mathematicians. The tool is meant to be used to iteratively refine a protocol design.

At the same time, human factors were very important to the tool's developers. The tool provides a common front end to a variety of back end formal methods engines. The front end consists of a "software through pictures" user interface, allowing the user to view the software as a set of interacting objects. The back end uses a version of belief logic whose libraries have been inspected on the source code level by members of the theorem–proving community. The back end receives the protocol specification through an intermediate language produced by the front end. Feedback is returned to the front end and displayed to the user.

To model a protocol, the user inputs a description consisting of beliefs, assumptions, and initial conditions from which the tool produces high level, graphical diagrams. The user can then see what goals are reachable. As a demonstration of the system, a somewhat simplified version of public key Kerberos was modeled. The usual problems in verification, insufficient initial conditions and wrong associations of messages and assumptions were experienced. The whole analysis took three days of tool use.

This tool provides a way to clearly specify the assumptions in a cryptographic protocol. It decreases the analysis time through automation, and it places formal methods in the designer's hands.

## Invited Talk

### Legal Signatures and Proof in Electronic Commerce

*Benjamin Wright, Attorney and Author – The Law of Electronic Commerce*

Benjamin Wright is a lawyer and the author of "The Law of Electronic Commerce." He started his talk by saying that his background is in law and society, and not in any scientific area. His talk would address, from a lawyer's point of view, an area that lies in the intersection of law and digital signatures.

Ben stressed that, when regulating uses of a technology, one needs to be humble and recognize that people may use the technology differently from how one has assumed. Different uses may require different legal interpretations and therefore require different legislations. So, he wanted to rename this talk "The Confessions of an Electronic Commerce Lawyer: My Fear of Curves on the Information Super Highway."

The law of signatures in the US has been around for a long time, and the legal community has been dealing with the question "what is a signature," for legal purposes, for centuries. Ben's own personal interpretation of "what is a signature" is liberal compared to that of some of his colleagues. His interpretation is: a signature, for legal purposes, is simply a symbol that someone adopts for the purposes of taking responsibility for a transaction. Generally speaking, the law of signatures in the United States has never required that signatures be secure in any way. This implies that signature, security, proof, and evidence are different things.

He then gave two examples of disputes that have happened with traditional signatures. The first example involved an autograph received through a fax machine. Its supposed signer argued that fax transmissions offered a low level of security, and the autograph was not his or her signature. The court ruled, however, that the lack of security did not interfere with the signatureness of the autograph. Authenticity, the question of whether or not the supposed signer signed it, was a separate issue and would be the one to look at.

The second example involved a real estate sale contract. The buyer backed out after sending a document that said: "I, X, agree to ..." At court, the buyer claimed that the document was not signed (no autograph was written at the bottom of the document). The judge, however, ruled that there was a signature in the document. The signature in this case appeared in the content of the message, where the buyer identified himself or herself as X. So, in fact, the document was signed, and the contract was legally effective.

Ben then went on to discuss what is currently happening with electronic signatures. Several states have been working on legislation regarding electronic signatures, and they don't agree with one another. Florida, Texas, Kansas and Iowa have adopted the liberal interpretation that Ben uses and separated the issue of signature from the issues of security, proof and evidence. He stressed that he personally likes this approach, but that there are disagreements in the legal community.

Other states, like CA, have adopted a more constrained approach. According to CA's Digital Signature Law, a digital signature is effective as a traditional signature if the signer has some kind of unique, verifiable code, if the code is under the signer's sole control, and if after the signer uses the code, one can determine if what was signed has been altered. (In addition, digital signatures need to comply with some state regulations that are still being elaborated.) So far, this legislation only regulates digital signatures involving the state of CA.

According to Ben, this view has advantages: it does not specify the technology, only the criteria. On the other hand, traditionally nothing has required that a signature has to achieve any degree of reliability or verifiability, so CA's legislation is less flexible in this sense.

Ben continued by talking about two completely different paradigms of electronic signatures, and the implications that their uses would have.

The first is the paradigm adopted by Utah's Digital Signature Act. Utah is a pioneer in regulating the use of public key cryptosystems in digital signatures. According to Utah's law, before I can use a private key as my signature, I need to go to a licensed certification authority (CA) to have my public key certified. After the certification, the private key has universal powers and I am responsible for keeping it secure. There is a presumption that any document that is signed with my private key is presumed to be my responsibility. It is not impossible to repudiate it, but it is difficult. This means that: 1) recipients of a signature have high evidence that the signer is going to be responsible for it; and 2) the owner of a key needs to be very careful with it, because the key can be used to sign any legal transaction, and the burden of proving the non-authenticity of the signature is on the person that owns the key. Ben's warning: "Don't let your spouse get it!"

Utah's digital signature law is an example of a concentrated transaction, where all the evidence is in one place: the signature. Dispersed transactions are the opposite of concentrated transactions. In dispersed transactions, the evidence that is needed for the receiver of a document to feel confident about it is dispersed across a number of factors and circumstances that are external to the signature. Previous relationships and the amount of money involved in the transaction are examples of such factors. When asked about Utah's failure to recognize the importance of other elements of a transaction, Ben said that he is concerned that the Utah law is too narrow and too detailed, and that there is no flexibility. However, he sees some areas in which signatures can be regulated this way. Electronic cash is one such example.

A second paradigm was then presented to contrast with the Utah approach. (Ben emphasized that he is not defending either approach, but is only giving us elements by which to judge for ourselves.) It is called PenOp, and uses the notion of dispersed transactions. It has been adopted by the IRS for electronic tax returns.

In PenOp, the taxpayer is shown the document he or she is about to sign. For the signature itself, he or she uses a digital pen to write on a digital tablet. The image of the signature, the speed with which it was written, and other biometric, "act-of-signing" data are then stored and constitute the signature. This biometric signature is then combined with the cryptographic hash value of the document, and the result is the signed document. For the IRS, having the document signed does not imply the document's non-repudiability. Instead, it is an evidence that you looked through your tax return and knew that you were legally responsible for it. In the case of a dispute, the IRS can not claim based only on the signed form that you were the person who signed it. Lots of other evidence is needed before they can prove that you were the signer.

Ben mentioned that the IRS+PenOp approach is probably not the Cypherpunks' favorite. Cypherpunks favor strong cryptography and oppose governments' intrusions into their citizens' private affairs. "What the IRS wants," Ben clarified, "is to get a little information about you, but not a whole lot. The security is mild, but things work in the context of other factors and circumstances." Ben said that for the IRS, security is a different issue and is ensured in different ways. For instance, the IRS has a private network to transmit its information.

An advantage of PenOp is that there are no keys, no passwords, and no training needed. One only writes one's signature.

Robert Gezelter asked about the shift in the burden of proof when adopting Utah's approach. Ben answered that there is in fact a shift. Traditionally, the receiver of a signature was the one to prove that the supposed signer actually signed it. Under Utah's approach, the signer is the one to repudiate it.

When asked about the method that UPS and FEDEX use, Ben said that their approach is less sophisticated, because only the bitmap of the signature is captured. No biometrics are used.

Someone asked what would happen if a PenOp signature is stolen and appended to another document. Ben said that the signature does not carry too much weight and its owner can easily repudiate it.

Ben Fried commented about the huge power that a signature has under Utah's approach and the ease with which this power can be transfered. Ben Wright agreed and said it is unprecedented in modern Western culture. He added that a private key is not a credit card. One can use private keys not only for financial transactions, but also for all other legal affairs like divorce, child custody, etc. One way to limit this power is to append disclaimers to the keys, restricting their use to specific purposes.

## Session III: Policy and Economics

### Digital Currency and Public Networks: So What If It Is Secure, Is It Money?

*John du Pre Gauntt, London School of Economics*

John posed the question "what is money?" He feels that electronic commerce is ready, or nearly so, from a technical standpoint, but that there has been insufficient consideration of what will "back" digital currency. He listed five properties of electronic markets (attributed to J. Yannis Bakos). One: the cost of communicating products and prices is lower. Two: the benefits of participation increase with the number of participants. Three: there are substantial initial switching costs. Four: they require large investments and benefit greatly from economies of scale. Five: potential participants are faced with uncertainties regarding the benefits.

He argues that given the importance of telecommunication for electronic markets, the market financial service providers will demand significant control over the telecommunications infrastructure and capacity. There are three types of money: the inherently valuable, that which has been directly backed by something inherently valuable, and money which has no direct value. Digital money is clearly not the first, and it is unlikely that states will initially give it the authority needed for the third, so what will back digital money? John suggested telecommunications bandwidth as a possibility. This concept raises many unaddressed issues. He detailed FLAG (Fibre-optic Link Around the Globe) as an example of the broadening connection between commerce and telecommunications. FLAG is a telecommunications cable which is principally funded not by telecommunications interests, but by investment houses.

While digital currency is being used on a small scale today, John argues that significant policy issues have been obscured by concern over the technical challenges.

### Modeling the Risks and Costs of Digitally Signed Certificates in Electronic Commerce

*Ian Simpson, Carnegie Mellon University*

The risks and costs of electronic commerce have not been sufficiently analyzed. Ian devised a quantitative model for the risks and costs of electronic commerce with certain parameters and assumptions. The certifying authority and the merchant are trusted. Cheaters are modeled by an initial compromise of the system, followed by chances for detection both over time and due to spending rate. This model was used to explore the state space of cheating, and enables a quantified assessment of risks. The model is still in a primitive form, and further analysis and refinement is expected.

Eric Bach asked if the participants in the model as described were behaving rationally. Ian responded that since the model is still in development, the participants don't always behave rationally.

## Session IV: Standard Payment Interfaces

### Generic Electronic Payment Services: Framework and Functional Specifications

*Alireza Bahreman, EIT*

Electronic commerce is becoming more popular. Increased demand is beginning to strain the infrastructure. The current solutions are not complete. The question is how we should build a framework for these applications to allow interoperability. One good approach is to identify the set of services necessary for electronic commerce and build these services and supporting infrastructure simultaneously. This work will focus on payment services, specifically transactions that involve peers who exchange values. This is only a subset of electronic commerce.

Alireza does not believe that SET should be the only solution. Payment models other than credit cards should be supported. Even if we only look at SET, it is likely that multiple vendors will add enhancements to it, so multiple solutions are almost unavoidable. The goal is to create a framework that will allow these solutions to interoperate. This will allow innovation, making it possible for new systems to be introduced easily. A framework would present an organized view of the confusion of payment systems to the user. And application developers would not have to rewrite their applications every time a new payment protocol was introduced. Instead, there would be a generic API for payment systems.

In GEPS, there are three layers: applications, services, and resources. Applications use services and services use resources. Any layer can change without affecting the other two layers. Applications come in two flavors: normal, which use services, and specialized, which are part of the payment infrastructure. Some examples of specialized applications are brokers for digital cash, banks, traders, and the government, which might back a payment scheme.

There are five different services: Transaction Management, Capability Management, Preference Management, Payment Method Negotiation, and Payment Interface Management. Transaction Management, used by the other services, is the most basic service. Its responsibilities include logging, maintaining transaction status, failure recovery, etc. Capability Management provides an interface between a user's services and various payment providers. It acts as a layer of abstraction between services and payment systems. Preference Management handles a user's configuration details, e.g. whether to use cash or credit, the maximum amount to spend per day, etc. It can be used to rank payment systems according to user–defined criteria. The purpose of Payment Method Negotiation is to negotiate which payment system will be used between peers. Finally, Payment Interface Management is an abstraction of GEPS for applications which do not wish to become embroiled in all GEPS' details. It presents a uniform interface consisting of configuration, a user's wallet, value transfer, and error handling.

Related work includes UPAI, IBM Zurich's Secure Electronic Market Place for Europe (SEMPER), and Java Electronic Commerce Framework (JECF).

### U-PAI: A Universal Payment Application Interface

*Steven P. Ketchpel, Hector Garcia-Molina, Andreas Paepcke, Scott Hassan, and Steve Cousins, Stanford University*

There is a diversity of payment mechanisms, all with different properties and different protocols. This diversity is a large problem for the application developer since applications will need to support all these protocols. All protocols have a general framework in common: the customer pays, some mechanism is invoked which handles the transaction, and the merchant receives the payment.

It would be useful to have an entity, called an Account Handle, that negotiates between the customer and merchant, handling details such as which payment system to use. The Account Handle would act as a proxy between a party and its actual commerce system–specific accounts. An entity known as the Payment Control Record would be used to control the transaction. It consists of a series of records of payment from one party to another. Each record would have transaction and control information. The customer and merchant would talk to the Record and the Record would talk to their Account Handles. Finally, entities known as Monitors will oversee the transaction and notify their owner, either the customer or merchant, through a callback mechanism if the status of a transaction changes. This removes the need for polling and gives the customer and merchant an entity to query about a transaction's status.

The system, which is object/method based, uses CORBA. Each entity has a set of methods which it can perform. Account Handles can create, open, close, or delete accounts as well as performing other maintenance tasks. The Payment Control Record can set the amount, source, destination, authorizations, etc. The Monitor, the implementation of which is left to the application programmer, has several types of status messages which it sends to the application: transaction completion, failure, or still pending. Additional programmer–definable subtypes of these messages can also be used.

This system can be adapted to support the delivery of electronic goods. The methods are the same, but the role of the customer and merchant are reversed. Instead of using an electronic commerce protocol, we would use an electronic delivery protocol.

Work in progress consists of adding more functionality such as support for pay per view, subscriptions, and shareware. Also, existing commerce protocols should be integrated into the system. Finally, security, which is currently absent, will be taken care of in future versions of CORBA.

In response to a question about what UPAI has to do with a digital library, Steve stated that the library needed a method of billing its customers for electronic payment services. Doug Tygar wished to know how this system related to a credit card payment via SSL. Steve responded that this would be just another payment protocol. Another person pointed out that UPAI does not present the user with all the options available in the First Virtual protocol. Agreeing, Steve said that while support for these options could be added, it would require application support as well. In general, the goal is not to support every foible of every protocol. Another question was whether only a few protocols would be in use eventually. Steve believes that there would be more than one protocol in use, so there is value in a common interface. Also, a common interface would make it easy to try new protocols.

## Payment Method Negotiation Service: Framework and Programming Interface

*Alireza Bahreman and Rajkuman Narayanaswamy, EIT*

Payment Method Negotiation is one of the five services of GEPS. The purpose of Payment Method Negotiation is for a merchant and a customer to come to an agreement on all details of the method of payment when they wish to transfer funds. The service deals with a variety of details about a payment, not just the choice of cash or credit. Some of the motivations for using negotiation are to make good payment method choices in a situation with too many options and to speed up the payment process through automatic negotiation.

Negotiation can take several rounds. In the prepayment stage, while a customer is shopping but before he wants to make a purchase, he could negotiate with the merchant just to make sure that there is a common payment method between them. This could be done in the background as the customer shops. When it comes time to pay, negotiation should be accomplished quickly. Negotiation could be symmetric, or one party could simply name a method. Then comes a finalization stage, in which the method is picked and the transfer of funds begins.

Related work includes the Joint Electronic Payment Initiative (JEPI), which addresses the protocol used to exchange messages. It includes the Universal Payment Protocol from Cybercash and the Protocol Extension Protocol over HTTP. JEPI is concerned with syntax and message flow, but it is not tied to a particular payment system. It is hoped that the payment method negotiation work will use JEPI for message syntax.

Negotiation is closely linked to the payment preferences and payment capability services in GEPS. It is assumed that both parties in a transaction have GEPS. There are three components in the implementation: input and output, payment negotiation, and a user interface. The user interface is specified, but left to the application to implement.

There are several policy issues. It is not clear who should or would want to reveal their capabilities. Also unknown is how long a negotiation will take. Finally, a method of picking the party who is to choose which payment system will be used has not been defined. Some policy decisions can be controlled by the way the application interacts with GEPS. Specifically, the application can control the order of negotiation, how many or few capabilities to reveal, how long negotiation will take, and in which phase the negotiation is. For further policy control, the GEPS objects can be subclassed.

Implementation of this system is in progress, though currently, the emphasis is on promoting the system. Alireza hopes that a very large and influential company will adopt GEPS as a standard, urging other companies to adopt it as well.

# Session V: Atomic Transactions

## Anonymous Atomic Transactions

*Jean Camp, Sandia National Laboratory; Michael Harkavy and J. D. Tygar, Carnegie Mellon University; Bennet Yee, University of California, San Diego*

Jean Camp began by pointing out that money sent over open networks is subject both to attacks and cheating by untrusted parties, and to network failures. She then defined several terms: money atomicity, which means that money is conserved in the system; goods atomicity, which means money atomicity plus guaranteed delivery of goods; and certified delivery, which means goods atomicity plus proof of precisely what goods were delivered. This discussion refers strictly to information goods, not to physical goods.

Jean went on to define anonymity as meaning that the identity of the consumer is not revealed, and to note that previous anonymous transaction protocols were not atomic: after the purchase both the customer and the merchant have the token and may be racing to cash it.

Jean stated that she would offer a protocol which was both anonymous and atomic, and give a proof by example. Her assumptions were: secure communication channels that don't reveal the consumer's identity; blinded signatures (Chaum) that enable signing of unseen data so that signature verification can't be linked to the initial signature; and a transaction log of messages from the customer, the bank, the merchant, and the log itself, which is an agent recording to publicly readable, reliable storage.

This is a two-part protocol. First there is a withdrawal or exchange of the token via a blinded request to the bank, which the bank signs; then the customer unblinds the request to get the token. The token is itself a public key, which the customer can prove has value, like a single-use certificate. Second, there is a purchase. At the customer's request, the merchant sends to the customer the goods in encrypted form along with a contract with the price and description of the goods. The customer sends an approval to the bank, the bank sends an approval to the merchant, and the merchant sends the encryption key to the log, which makes it publicly available.

The log is the transaction coordinator, and issues either a global commitment of the transaction or a global abort. We assume that the merchant trusts the transaction log not to release the key without issuing a commitment, and that the consumer trusts the transaction log to publish the key in a timely manner. The bank and the transaction log could potentially be combined – separation helps to minimize how much each party needs to be trusted. The protocol is anonymous because each token is a public key that is not linked to the customer's identity.

Jean then discussed some possible variations. Reusable consumer keys would be more efficient, but less anonymous as they would allow the bank to link a series of transactions. Cryptographic timestamps in the log would help to reduce the risk of delay and minimize the damage if the log were compromised. Encrypting the log would get rid of observers. Two-sided certified delivery would allow the merchant to take the burden of proof.

Questions for future development include how to make the protocol more efficient and more scalable, and how to reason formally about anonymity. Policy issues relevant to this work include legal requirements on transaction sizes and aggregate data collection, and key escrow.

Dan Geer asked Jean to elaborate on how she sees the role of anonymity, and whether it is possible to achieve privacy through other means. Jean answered that it is possible to get privacy through policy mechanisms to some extent, but that this is subject to violation since you have to trust others to maintain your privacy. Jean was asked if any prototype of this system has been built, and replied that none had, nor had any analysis been done, and that public key systems are generally expensive. Doug Tygar then added that storage requirements would be quite large as they are for all digital cash systems which must keep log and bank records.

## Strongboxes for Electronic Commerce

*Thomas Hardjono and Jennifer Seberry, University of Wollongong*

Thomas Hardjono suggested that an electronic counterpart to physical strongboxes could be a useful mechanism for electronic commerce. These electronic strongboxes would be used for secure storage of anything digital, including certificates, contracts, cash, coins and checks. Access to an electronic strongbox service would ameliorate the need for users to have large storage spaces personally for storing all their electronic cash, and would facilitate trade and exchange.

Thomas further suggested that physical valuer services might generate unforgeable digital representations of the value of existing physical goods. With the real goods in secure physical storage, on-line valuers could be used to verify the authentic ownership of items in the system. to split items into sub-items, and to help the exchange facilitator, who would mediate exchanges between customers to ensure that they were honest and irrefutable. A formal association would oversee the entire system and its participants, and a notary would handle disputes in cooperation with the association.

Such a system would require proof of the retrieval or storage of an item in a strongbox, proof of the ownership of an item with anonymity, proof of the submission of an item for valuation, proof of an exchange transaction, and detection of illegal and duplicate items.

Bob Gezelter asked if this implied that a user would in effect create a shadow currency by depositing something of value; Thomas answered that yes, it did. It was then pointed out that anonymously tradeable certificates equal cash.

## Model Checking Electronic Commerce Protocols

*Nevin Heintze, Bell Labs; J. D. Tygar, Jeanette Wing and H. Chi Wong, Carnegie Mellon University*

Chi Wong outlined an electronic commerce system consisting of a customer, a merchant, a bank, and goods that can be sent over a network. She then defined two main properties of the system: money atomicity, which requires that the total money in the system remains constant; and goods atomicity, which requires that the customer receives the goods if and only if the merchant receives the money. She described the possible failure modes of the system as either network or processor failure, and cheating, which would be an attempt by the customer to double-spend, or an attempt by the merchant to double-deposit.

Chi then explained that model checking, an approach based on exhaustive search of finite state spaces, could be applied to this system to verify its properties. A model of this system and a property specification could be given as input to a model checker, which would return a yes, meaning that the properties were verified, or provide a counterexample.

In FDR model checking, which stands for "Failures Divergence Refinement," the system model and the property specification are both state machines represented in the same language. The model checker then implements a refinement relation to see if the state space given by the model is a subset of the state space given by the property specification. Chi described building FDR models of simplified versions of the NetBill and Digicash systems, which were then run through a model checker; she referred the audience to the paper for the results, noting that while model checking has been useful for hardware verification, and recently also for software verification, this is the first time it has been applied to electronic commerce protocols. As future work, Chi hopes to create a more complete failure model, do more complex runs, add more properties, and explicitly represent the role of cryptography, which is currently abstracted away.

Dan Geer asked whether it was possible to use these techniques during design rather than just analytically; Chi thought probably not with FDR model checking, but that automatic program generation refinement tools might be useful. Eric Hughes asked how the size of the state space affected the process; Chi referred him to the appendix, noting that the example given had about 3000 states. Eric then asked about asymptotic properties, and Chi answered that the model checker looks at all possibilities and thus is exponential; Eric asked if symbolic modeling had been considered as a technique to cut it down, and Chi replied that it had, and that a tech report would follow.

Jeanette Wing then followed up on Dan Geer's question by stating that yes, you can use these model checkers for iterative checking and design of protocols, and that it would be pushbutton technology so the iteration would be fast. She noted that SMV is a richer language for expressing properties, and that FDR is tuned to check deadline detection and is more limited.

# Session VI: Experience

## BigDog: Hierarchical Authentication, Session Control, and Authorization for the Web

*Benjamin Fried, Andrew Lowry, and Morgan Stan-*

*ley*

The goal of BigDog is to use the WWW to interact (e.g. deploy applications, exchange data) with existing clients, not to recruit new clients or establish new relationships. BigDog incorporates different levels of security to accommodate different levels of data sensitivity. The SSL is used to encrypt all data flow. "Home site" (i.e. IP address) information is also used. An access control list is maintained on a per use, per resource basis. The work evolved, and Ben stated that their experience is that plug-ins were troublesome. The model of separate communicating protocols provided more freedom and worked better. He mentioned that this work is related to OM-Access.

Ben was asked if user input was used to design BigDog. He answered that some input was used, but commented that users are not necessarily educated about security issues. Eric Hughes asked what plans there were for risk analysis. Ben indicated that it would be nice to be able to indemnify to auditors. Bob Gezelter asked why the IP address information was used, since it can be spoofed, to which Andrew responded that it was used as only a minor security component – a "half step" in security. Steve Jones asked how users or resources were grouped. Andrew said that that such grouping took place in the administration, not in BigDog itself. When asked, Ben said that it was hard to estimate exact costs, but that around three person months went into the project. Ed Uielmetti asked if there was support for out of band authentication. Andrew replied that one application does use it. In response to a question from Andy Rabagliati, Andrew stated that the system operated on SunOS/Solaris.

## Financial EDI Over the Internet: Case Study II

*Arie Segev, Jaana Porra, and Malu Roldan, University of California, Berkeley*

Bank of America, for which this work was done, is a multi-billion dollar corporation and operates in 36 countries, so size and scale presented extra challenges to the project. The planning involved meetings of many people from many different fields. The project was approached as a learning experience. Privacy Enhanced Mail (PEM) was used to provide security. During phase 1 of the project, the scale was kept small. No messages were lost and no tampering was detected, but very few transactions were processed. After this early success, the limits and volumes were increased, and a more substantial number of transactions were processed without detected tampering. Testing showed that the FEDI system (rather than the network) caused most of the delays in messages. Public perceptions of security are an obstacle to this sort of project.

Dan Geer asked if a large part of the problem was from the lack of value added networks; Jaana replied "yes". Dan followed by noting that VANs are a small part of the total cost of existing financial transactions. Jaana responded that this is true, and that the cost benefit analysis for this approach is not entirely clear. Bob Gezelter commented that over-batching of messages could be responsible for much of the latency, and Jaana agreed that this was possible.

## Scalable Document Fingerprinting

*Nevin Heintze, Bell Labs*

In part as a response to being plagiarized, Nevin became interested in finding a scalable means to catch copied (or partially copied) papers. The fingerprinting should be sufficiently robust to catch moderate variations of the same paper. A sliding window of chunks of a certain length (e.g. 30 characters) is used to go through the document, creating a fingerprint against which other documents' fingerprints can be matched. Each document's full fingerprint can't be stored if there are many documents, so the method used was to keep a fixed number (say 100) of chunks from each document in the database. In order to reduce false positives infrequently occurring chunks were used in the fingerprints. This was combined with hashing to select certain chunks and increase matches.

The method worked well when tested on a corpus of CMU technical reports and on a larger corpus of reports available electronically. Even a one percent match indicated a likely similarity. In practice, techniques such as ignoring headers, introductions, references, and other highly repetitious information helped to reduce false positives and provide more precise detection. In order to defeat an iterative attack, in which a plagiarized document is repeatedly modified until the matches are eliminated, occasional random resamplings should be performed, updating the chunks associated with each document in the database.

# Lunch with Invited Speaker

## Designing New Rules of the Road for Electronic Commerce in Digital Information

*Pamela Samuelson, University of California, Berkeley*

Currently, there are three new sets of rules being proposed for the information superhighway. One set is about the validity and meaning of contracts, another is on revisions to copyright law, and a third concerns new intellectual property law for database contents. All three areas will change the law (both national and international), strengthening the rights of information vendors. These changes are being made to encourage the electronic commerce market and to help United States information providers continue to dominate the market. These rules are close to being implemented.

One of the proposed laws would validate the terms of "shrink wrap" licenses. These are traditionally debatable since a licensee does not see the license until after the shrink wrap has been broken. A federal court ruled in favor of this view, finding that a shrink wrap license was indeed invalid. However, an appellate court overturned this ruling, stating that the license was valid and could be enforced if a user continued to use the product after seeing the license. This decision implies that a license to use software, rather than ownership of a copy of the software, can be sold. If a person violates the license, then that person loses the license to use the software.

Another change is coming in the area of implied warranties. Currently, an implied warranty basically says that a product should work. But under the proposed changes, unless there is an explicit statement of quality, the implied warranty would be that the manufacturer did its best to make the software correct.

In the area of copyright law, a large expansion over the control of reproduction has been proposed. Today, a copy has to be "tangibly fixed" to be considered an infringement. In the future, temporary copies, even caching an image in RAM, could be considered an infringement and thus can be controlled by the owner of the original. Any digital transmission of an object would be considered a communication of the work to the public and could be controlled by the object's owner. This would mean that, contrary to traditional copyright law, a person can be restricted from giving a document to a friend when he no longer wants it. Also, web crawlers, since they keep temporary copies of documents, would become illegal.

All of these proposals are part of a Clinton administration white paper. The strategy behind these proposals is to garner international support, which will force these changes to be adopted in the United States. Congress would not need to adopt any international treaties that are proposed, but could simply implement equivalent legislation.

In the area of database content, there is a proposal to grant the producer of a database, if a significant amount of effort was invested in creating it, 15 or 25 years of exclusive control over the extraction of information from the database without exceptions for fair use or research. The goal of this proposal is to protect the United States database industry against people who pirate information.

Bob Gezelter predicted that not allowing caching would drive the Internet into the ground with high load. He wondered whether there would be an exception for delivery mechanisms. Pam answered that if the Clinton administration had thought about delivery mechanisms, they would probably view them as infringing. Another opinion is that caching is an aid to people who deliver information, so under fair use, it should be acceptable. But in the Clinton administration white paper, caching in RAM was specifically mentioned as a cause for infringement. Another person asked whether these proposals were sparked by "banality or stupidity." Pam replied that the motivation was to protect the entertainment industry, but that it's time we moved away from the attitude that something that's good for a specific business is good for the whole country.

One question was whether everything on the World Wide Web would need to explicitly state what rights were granted to users. Pam replied that the proposals would change the ground rules and we would no longer be able to assume rights such as keeping temporary copies. One person wondered whether people would own their personal information, such as data traditionally used for marketing. Twenty years ago, Pam replied, the answer would have been a flat "No." Today and in the future, information is becoming more like property. In fact, the proposed database treaty would make information into property.

Another question was whether a web site with many links to other sites could be considered a database. Pam said that this issue has not been addressed yet, but the database bill in the House of Representatives defines a database as a collection of information materials arranged in a systematic way. In addition, copyright law would still apply to the content of the web site.

One person was curious about whether the Europeans would have fair use exceptions to copyright laws. Pam responded that there are several rules about this. In one approach, users would have the right to take insubstantial parts. Another approach would be to disallow taking any part, no matter how small. The last approach is that a piece, even a substantial one, could be extracted for illustrative purposes such as education, but not for analysis. The drive for new database legislation was unknown at first in the scientific and education communities. Recently, these communities are coming out against the new legislation.

One person wondered whether we could use cryptography and fair competition laws to take care of concerns about people copying CD-ROMs. Pam replied that most violations of the proposed laws would already be violations of current laws. The goal is to stop a slight leakage from becoming a hemorrhage.

## Session VII: Protocols

### A Protocol for Secure Transactions

*Douglas H. Steves, Chris Edmondson-Yurkanan, and Mohamed Gouda, University of Texas, Austin*

Douglas Steves opened the session on protocols. He and his co-authors are interested in secure transaction protocols as a means of achieving secure electronic commerce. They proposed a protocol with strong relational properties.

Doug started by contrasting secure communication protocols with secure transaction protocols. In secure communication protocols, the main concerns are privacy, authentication, integrity and non-repudiation. PGP and PEM are the best known examples of this class of protocols. SSL, SHTTP and SET, although they have the notion of sessions, do not establish relationships between the messages in a session, and are therefore considered examples of secure communication protocols.

According to Doug, message security properties are not enough for secure transactions. Relational properties that link the multiple actions in a transaction are crucial. He then identified three relational properties: atomicity, isolation, and causality. They are all present in the standard database (DB) theory and legal contracts. Atomicity and isolation have been discussed by Tygar and his colleagues, but causality is new here. Basically it says that it is not enough for two (or more) messages to be part of the same transaction; the ordering of these messages is important.

At this point, he opened a parenthesis and said that secure transaction protocols should lie underneath electronic commerce protocols. The question of role playing (who is the customer and who is the merchant), as well as forms of exchange media (credit cards or electronic cash) should all be part of this higher level. Close parenthesis.

Returning to the main focus of the talk, Doug said that the way that he looked at atomicity was different from the way that Tygar looked at it. Atomicity, for Tygar, appears in a concentrated form: both the commit and exchange of goods and money takes place in one point in time and space. Their view of atomicity is dispersed: commit and exchange are physically and logically separated, the exchange being dependent on the commit.

With respect to isolation, Doug Steves and his colleagues' definition is that all or none of the transaction messages are valid. In DB operations, isolation is guaranteed by the DB manager, which only allows the result of a transaction to be seen by the outside world when the transaction is complete. In a message exchange system, isolation is hard to guarantee, since messages sent over the network can be caught, copied and stored at will.

Causality was first discussed by Lamport, who introduced the notion of vector stamps to indicate the ordering of messages in distributed systems. Under this approach, the receiver of a message only knows the number of messages that have been sent and received previously by the sender, but does not know the contents of the messages. Authentication was added to vector stamps by Tygar and Smith. In 1993, Ken Birman proposed piggybacking messages on top of other messages, thus introducing a new form of causality where one can talk about the contents of previous messages. In 1996, Gong and Reiter combined Birman and Tygar and Smith's proposals and obtained secure causality. It is this form of causality that Doug Steves uses in his transaction protocol. Thus, when committing to a transaction, the protocol commits to the messages of the transaction and to the order of the messages in the transaction.

The protocol that Doug Steves and his colleagues have implemented proceeds in three phases (initiation, exchange and termination) and uses half-duplex communication. Atomicity and isolation are achieved via the two-phase commit mechanism, and causality is achieved via Gong and Reiter's mechanism.

During the question period, someone stated that SET also satisfies isolation, atomicity and causality. When asked how his protocol differs from Gong and

Reiter's, Doug said that Gong and Reiter did not address atomicity and isolation.

## PayTree: "Amortized-Signature" for Flexible MicroPayments

*Charanjit Jutla, IBM; Moti Yung, Banker's Trust*

Charanjit Jutla presented the PayTree payment mechanism. He started with a summary of the major steps in micro-payments systems and pointed out that it is crucial to make payments from customers to merchants computationally efficient.

Public key signatures are the most reliable way of authenticating or verifying payments, but they are computationally expensive. The idea is therefore to minimize the number of public key signatures that are required in issuing or authenticating (a sequence of) certificates for payments.

Charanjit briefly explained the workings of PayWord, a scheme that explores this idea. Based on Lamport's one-time password scheme, PayWord only requires one public key signature to issue a number of payment certificates. By linking the validity of future payments to the validity of a previous payment through cheap hash functions, the cost of the signature operation is amortized.

PayWord, however, is not able to determine who the cheater is when fraud occurs. For instance, if a certificate is presented more than once for redemption, the bank does not know if the customer double spent it, or if the merchant colluded with another merchant. Also, it is not well suited for web surfing, because payment certificates generated by a signature can not be spent with different merchants.

The idea of PayTree was then presented. It is based on Merckle's authentication tree scheme, and uses the following data structure: a tree whose leaf nodes are labeled by secret random values, whose internal nodes are labeled by the hash value of the nodes' successors, and whose root is signed. Because of the tree structure, PayTree is more flexible and allows payments to different merchants to be made using different parts of the tree. This means that multiple merchants can now share the cost of a public key signature. Charanjit proceeded to describe ways of issuing and verifying payments in three different scenarios. In the first scenario, a tree is used to pay only one merchant; in the second scenario, a tree is used to pay multiple honest merchants; in the last scenario, multiple merchants with arbitrary behaviors are considered. The computational complexity of each of the cases is presented.

In the basic PayTree mechanism, the individual payments all share the same value and each tree has a pre-defined total value associated with it. But PayTree can be slightly modified to implement trees with multiple denominations, unlimited payment potential or divisible coins. It can also be used as a module of other payment systems.

Someone in the audience commented that the flexibility of PayTree increases bandwidth and storage consumption. Charanjit agreed.

## Agora: A Minimal Distributed Protocol for Electronic Commerce

*Eran Gabber and Abraham Silberschatz, Bell Labs*

Eran Gabber presented Agora, a micro-payment protocol that relies on the assumption that public key signatures are not that expensive. The goal was to design a micro-payment protocol that is compatible with existing tools and protocols, scalable and distributed, and whose overhead per transaction is minimal. The overhead that matters is given by the number of messages and signatures required by the protocol for a typical transaction. Micro-payments that have been proposed so far are not designed with the number of messages in mind. They are not concerned with compatibility with existing web protocols either.

Agora fulfills the requirements mentioned above. It is minimal: a typical transaction does not generate more messages than what is required for web browsing. It is distributed: a typical transaction can be verified by the merchant without contacting any online authority. It enables online purchases: one does not need to go to a broker first to get scrips.

Eran went on to describe the protocol. There are five kinds of entities: a central authority, who certifies the banks' public keys; banks, who manage accounts; customers; merchants; and arbiters, who also need to be registered with the central authority. Both customers and merchants have accounts (with expiration dates) at the bank. The expiration date helps in housekeeping billings and payments, and lessens the effect of brute force attacks. A billing period is associated with the lifetime of each account. Everybody has public keys and private keys. New sets of keys are generated for each billing period. Whenever the merchant starts with a new pair of keys, he or she takes them to the central authority for certification. When customers change their keys, they go to their banks to get new certificates for the next billing period. A certificate is a customer ID, signed by the customer's bank, and includes the expiration date, an account number and the customer's public key. The certificate is used as a promise of payment. Banks would only issue new certificates to customers that paid their bills for the previous period.

For a typical transaction, the protocol uses four messages. The first message has the customer asking for price quotations; the second message has the merchant reply with the quotations; the third message contains the purchase order from the customer; and the last message, from the merchant, contains the goods (or error messages). This is the same number of messages that free web browsing takes. The first message is generated when the customer clicks on a link containing a page of price quotations; the page is displayed to the customer when the second message is received; the third message is generated when the customer clicks on an specific item; the item is then returned in the last message and displayed to the customer.

Note that the merchant sends more quotations than have been asked for. This is an advantage, because if the customer decides to purchase a second item on the same page of quotations, only two message are needed in the transaction. Also, only one signature is used for each page of quotations.

Because it is assumed that merchants distrust banks with whom they have not had relationships before, the first time the merchant receives a certificate issued by a bank that he or she does not know, the merchant may go to his or her own bank and verify the unknown bank's public key. (Every bank's key certification is broadcast to all the other banks by the central authority.)

Merchants can therefore accept certificates in a distributed fashion, and only submit them at the end of each billing period. This offline mechanism does allow merchants to be defrauded in situations where they accept certificates from accounts that have been revoked or expired.

Eran continued with a security analysis. Because all messages are signed, authenticity, tamperproofness, and non-repudiability are guaranteed. Because all messages come with sequence numbers, replay attacks and double charging by merchants can be prevented. But, as mentioned before, full distribution makes fraud possible in this protocol. To lessen the possibility of fraud, it is possible to enhance the protocol and have merchants talk to customers' banks now and then. (Banks need to keep track of all revoked certificates.)

If the customer sends a purchase order, which constitutes a promise to pay, and never gets anything back or gets something else, then he or she can go to the arbiter and present the quotation (which includes the description of the goods) and the purchase order. The arbiter can then demand the goods. If the merchant does not comply with the demand, the arbiter has the power to revoke the transaction.

During the question period, Mark Manasse asked about the impact of doing digital signatures on the latency of transactions. Eran replied that the customers' machines are assumed to be idle, so there is no problem there. Merchants, however, should have a farm of PCs dedicated to signature generation and verification. Also, one can always use optimized signatures to make things more efficient. Marvin Sirbu commented that Agora reduces the number of messages by transferring liability to the merchant. Eran agreed.

# Panel Discussion

## Electronic Commerce in Practice – What Have We Learned?

*Clifford Neuman, University of Southern California (moderating); Ed Vielmetti, First Virtual Holdings, Inc.; Marc Briceno, DigiCash; Steve Crocker, Cybercash; Daniel Geer, Open Market, Inc.; Malu Roldan, University of California, Berkeley; David Van Wie, InterTrust*

The members of the panel each spoke briefly about their experience with building electronic commerce systems in the real world. Steve Crocker described Cybercash's automation of the customer/merchant payment authorization process, and said that they have about 150 merchants using their main system, and about 20 merchants using their newer small payment system. Dan Geer stated that OpenMarket has moved its focus to automating web commerce between businesses, rather than between merchants and customers. Malu Roldan described how existing companies move toward using the web, saying that there's a lot of interest in cheap initial experimentation. David Van Wie said that InterTrust is focusing primarily on information commerce, distributing movies, newspapers and software electronically, and that InterTrust also believes that business to business is the place to start. Ed Vielmetti described First Virtual as a global, low cost payment system in which anyone can be the buyer and anyone can be the seller, with about 200 merchants and 180,000 customers. Finally, Marc Briceno described Digicash's e-cash system for providing the buyer with complete anonymity.

General discussion and questions from the audience followed. Some points of agreement seemed to be: that customer service actually tends to be a bigger cost than fraud, at least for retail systems; that our understanding of what we want in an electronic store is still evolving; that there is plenty to be done in automating existing real world systems; and that merchants are looking to reduce costs more than to increase business. Opinions were divided on the usefulness of risk modeling. The difficulty of building global electronic commerce systems while different countries have different laws and expectations was acknowledged.

# Session VIII: Security

## Organizing Electronic Services into Security Taxonomies

*Sean Smith, IBM Research; Paul Pedersen, Los Alamos National Laboratory*

As the world moves to depend more on electronic services, it is desirable to have a method to analyze the tradeoffs being made. We wish to know the vulnerabilities and points of attack of any given system. Sean suggests a structured approach, building a taxonomy of the vulnerabilities from the inherent structure of the provided services. They placed a partial order on the various services which can be provided, and looked at the differences between the two steps. Services inherit vulnerabilities from below (i.e. weaker services), and stronger services can introduce new vulnerabilities as well. This taxonomic structure also works to model points of attack, which can be thought of as "inadvertent services." An example case is kiosks. There were difficulties resolving the levels of services into quantum steps. A variety of properties (e.g. spatial extension, input privacy) were used to describe the provided services and build the structure of vulnerabilities. Sean emphasized that this is a prototype, and that it is being refined and extended.

Doug Tygar asked if there were hopes for making the process more general. Sean said that the system has some generality, more than shown in the example. Ed Uielmetti asked about weaknesses that are the result of combined services, and are not weaknesses in the components. Sean responded that this is not covered by the method, but that work is in progress.

## WWW Electronic Commerce and Java Trojan Horses

*J. D. Tygar and Alma Whitten, Carnegie Mellon University*

Alma brought up ways in which WWW commerce can be attacked that are based on the way people browse the web and weaknesses in the security model. The attacks presented do not rely on implementation faults, but rather are weaknesses in the way the system is designed. The two attacks presented are bogus remote pages and local Trojan horses.

The bogus remote page attack relies on the lack of verification of who operates a particular page or electronic storefront. Users do not usually check address names, and domain names are available that could be used to plausibly impersonate a real site. Given the ease of copying electronic information, an attacker simply creates a site which looks like a trusted site. When the user's browser is pointed to the bogus page address, an applet which spoofs the trusted page takes control, and thus the bogus remote page enables the local Trojan horse attack.

Once the attacker applet has control, it can spoof secure dialog boxes and act like the spoofed site while obtaining potentially sensitive information (such as a password or credit card number) through the user's entries. This information can be sent back to the attacker's site by hiding it in the page access requests. After this is done the applet passes control to the real site, and the attack goes unnoticed.

Code signing is not a sufficient fix for this problem, since it requires a trust basis and it will still be desirable to run unsigned applications, since code verification is expensive. A solution is window personalization. Make the trusted aspects (such as the background of the dialog boxes) 1: distinctive and easy to recognize for the user and 2: difficult for a prospective attacker to predict. To make this method work, Alma suggests the following: require selection at installation, educate users, offer many choices with randomized defaults, and avoid company logos or other predictable designs. There are extensions of this approach to ATM and POS applications.

Alma was asked why location couldn't be used to indicate genuine dialog boxes, and responded that pages may occupy the entire display. Bob Gezelter mentioned the importance of good randomization, and Alma agreed. Alma was asked why the local Trojan horse was necessary. She replied that the local Trojan horse attack is more general than just the bogus remote page. Ben Fried suggested that code signing with trust determined by the vendors would alleviate the problem. Alma said that even if trust assumptions were given, code signing is inherently subject to potential flaws.

### On Shopping Incognito

*Ralf Hauser, McKinsey Consulting; Gene Tsudik, University of Southern California*

Gene presents a system for anonymously performing electronic commerce. The system involves four phases, browsing, obtaining offers, payment, and delivery. Privacy is important in all phases. Identity information can be used, for example, in junk mailing lists or unfair pricing. It is important that transactions be unlinkable. In the first phase, prepurchase browsing, the consumer collects signed offers of price and description, which may or may not be transferable. Gene presented two protocols, prepurchase browsing (PPB) and electronic merchandise delivery (EMD). The PPB protocol supports anonymous browsing, but identity information is revealed during delivery. EMD supports anonymous merchandise delivery. They can be combined to form a completely anonymous system. The protocols provide signatures to the participants which enable them to prove in court what transpired. Gene referred to this as a cop out.

Eric Hughes asked why Gene considered the court system a cop out. Gene clarified that he simply means that actual court involvement would be very costly, but that, as in paper transactions, the backing of the court system is necessary. Andy Rabagliati mentioned that the wide prevalence of transatlantic caching would help support privacy, and Gene agreed that it would help support browsing. In response to a question, Gene indicated that merchants would want to support this to provide for their customers, and that it might have applications in situations of political oppression as well as the obvious application in pornography. Bob Gezelter mentioned that upcoming copyright legislation might interfere with some of this protocol.

## Session IX: Software Agents

### Market-Based Negotiation for Digital Library Services

*Tracy Mullen and Michael P. Wellman, University of Michigan*

Tracy Mullen presented work on market-based negotiations for digital library services. This work is based on two assumptions: 1) available resources are limited, and 2) what people may want to do in a digital library is unpredictable. Given these assumptions, digital libraries should provide a flexible, open and extensible infrastructure that supports different market practices.

The University of Michigan Digital Library (UMDL) project is designing and implementing a digital library based on a system of software agents that interact with each other and with end users. Software agents are used to perform various activities needed to deliver goods and services. Because there are multiple competing agents trying to accomplish multiple tasks as efficiently and as cheaply as possible at a given time, resource competition needs to be resolved. UMDL sets the agents to negotiate with each other, so that globally optimal agreements can be reached. Instead of hardwiring pre-defined negotiations, UMDL uses user-definable auctions that can be dynamically established when goods or services are to be sold or bought. Auction mechanisms need a number of parameters to be fully specified. These parameters include the type of goods, price quote policy, price quote interval, clearing policy, and tie breaking, among other things.

UMDL's digital library is a market place with dynamically evolving configurations that include goods that are being sold and the mechanisms by which they are negotiated and exchanged. When given buyers' demand profiles and the current resource congestion profile, the system will decide on a level of service for each buyer. Currently, there is an auction server working.

During the question period, someone asked about the danger of shill-bots. Tracy answered that it is possible to use certificates of "honesty" to decide who is allowed to transact in the system. Christian Frank wondered about the scarcity of information goods. Tracy pointed out that one should look at other resources as well. Users' time and the system's computational resources can both be scarce. Someone asked about protections that the system offers against unexpectedly high demands. Does UMDL prevent crashes from happening in such scenarios? Tracy answered that the marketplace is self-regulating. As the demand increases, the price of the goods or service also increases, which can drive the demand down. But UMDL can also resort to distributed auctions. The last comment came from Doug Tygar: "The notion of having auction markets for information goods is terrific, because that means that we may have a futures market and I can short my colleagues' papers!"

### Information and Interaction in MarketSpace – Towards an OpenAgent-based Market Infrastructure

*Joakim Eriksson, Niclas Finne, and Sverker Janson, Swedish Institute of Computer Science*

Joakim Eriksson's talk described an agent-based infrastructure that he and his colleagues are building to automate market interactions, such as searching for business partners, negotiating, and settling a deal. Although the Web can be used for such purposes, it is not quite adequate because: 1) the data on the Web is unstructured (there is text, graphics, video, etc.), and 2) the type of interaction offered by web browsers is not tailored to business transactions.

The main focus of the work is to develop information and interaction models. The information model should satisfy the following requirements: 1) the data should be presented as structured knowledge; 2) the participants' interests and their potential business deals should be adequately represented by well-defined description languages; and 3) the approach should be object-oriented. With respect to the interaction model, it must be simple, but able to model a wide range of types of market interactions. Examples of primitives one can use to carry out an interaction in their model are: ASK, TELL, NEGOTIATE, OFFER, ACCEPT, and REFUSE. Joakim then showed how information is represented and how interactions are modeled under their approach through a number of examples. The examples included direct transactions between two individuals, someone seeking the help of a broker, and an auction.

Joakim then briefly talked about other components of Market Place, the project that the work in this paper is part of. One of the components connects Market Place with the Web, allowing users of one system to use the other. The Market Place group is also developing agents that have roles (brokers, auctioneers, buyers, etc.). The system will be tested in Spring 1997.

During the question period, Eran Gabber asked about the use of English as a universal language in their system. Joakim said that the Web also has this problem, and that the Market Place is not trying to solve it. Instead, they plan to just plug in solutions, once they are available.

## A Peer-to-Peer Software Metering System

*Bruce Schneier and John Kelsey, Counterpane Systems*

Bruce Schneier was delayed by bad weather and the presentation of this paper was canceled.

# THE USENIX ASSOCIATION

Since 1975, the USENIX Association has brought together the community of developers, programmers, system administrators, and architects working on the cutting edge of the computing world. USENIX conferences have become the essential meeting grounds for the presentation and discussion of the most advanced information on new developments in all aspects of advanced computing systems. USENIX and its members are dedicated to:

- problem-solving with a practical bias
- fostering innovation and research that works
- communicating rapidly the results of both research and innovation
- providing a neutral forum for the exercise of critical thought and the airing of technical issues

### SAGE, the System Administrators Guild

The System Administrators Guild, a Special Technical Group within the USENIX Association, is dedicated to the recognition and advancement of system administration as a profession. To join SAGE, you must be a member of USENIX.

### Member Benefits:

- Free subscription to *;login:*, the Association's magazine, published 6-8 times a year, featuring technical articles, system administration tips and techniques, practical columns on Perl, Java and C++, book and software reviews, summaries of sessions at USENIX conferences, Snitch Reports from the USENIX representative and others on various ANSI, IEEE, and ISO standards efforts.
- Access to papers from the USENIX Conferences and Symposia, starting with 1993, via the USENIX Online Library on the World Wide Web.
- Discounts on registration fees for the annual, multi-topic technical conference, the System Administration Conference (LISA), and the various single-topic symposia addressing topics such as object-oriented technologies, security, operating systems, electronic commerce, and NT - as many as ten technical meetings every year.
- Discounts on the purchase of proceedings and CD-ROMs from USENIX conferences and symposia and other technical publications.
- Discount on BSDI, Inc. products.
- Discount on all publications and software from Prime Time Freeware.
- 20% discount on all titles from O'Reilly & Associates.
- Savings (10-20%) on selected titles from McGraw-Hill, The MIT Press, Morgan Kaufmann Publishers, Sage Science Press, and John Wiley & Sons.
- Special subscription rate for *The Linux Journal* and *The Perl Journal.*
- The right to vote on matters affecting the Association, its bylaws, election of its directors and officers.

### Supporting Members of the USENIX Association:

ANDATACO
Apunix Computer Services
Auspex Systems, Inc.
Cirrus Technologies
CyberSource Corporation
Digital Equipment Corporation
Earthlink Network, Inc.
Hewlett-Packard India Software Operations
Internet Security Systems, Inc.
Invincible Technologies Corporation
Lucent Technologies, Bell Labs
Motorola Global Software
Nimrod AS
O'Reilly & Associates
Performance Computing Magazine
Sun Microsystems, Inc.
TeamQuest Corporation
UUNET Technologies, Inc.
WITSEC, Inc.

### SAGE Supporting Members:

Atlantic Systems Group
Collective Technologies
D.E. Shaw & Co.
Digital Equipment Corporation
ESM Services, Inc.
Global Networking and Computing, Inc.
Great Circle Associates
O'Reilly & Associates
Remedy Corporation
Sysadmin Magazine
TransQuest Technologies, Inc.
UNIX Guru Universe (UGU)

For further information about membership, conferences or publications, contact: USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710 USA. Phone: 510-528-8649. Fax: 510-548-5738. Email: *office@usenix.org*.
URL: *http://www.usenix.org*.

ISBN 1-880446-97-9